Workflow Management

Software Test Report

Submitted in partial fulfillment of the requirements of the course CS223 – Software Engineering

Submitted by

Chitraksh Sadayat (B16CS007)

Vishakh Suresh (B16CS038)

VERSION HISTORY

Version #	Revision Date	Brief Description
1.0	29/03/2018	Unit testing and Integration Testing

TABLE OF CONTENTS

1. INTRODUCTION …… 4
   - Purpose …….. 4
2. TEST PLAN ……..… 4
3. DETAILS OF UNIT TESTING …… 5
4. DETAILS OF SYSTEM TESTING …….. 57
5. APPENDIX (Control Flow Graphs) …….. 65
6. APPENDIX (Glimpses into the execution) …… 78

1 INTRODUCTION

1.1 PURPOSE

This *Workflow Management Unit and System* Test Report provides a summary of the results of test performed as outlined within this document. Testing is done to eliminate faults and errors to avoid user-end failure of the software.

Tests are performed under following specifications:

1. **In-scope**:
   The functionalities of following modules are tested:
   a) Apply leave: for Undergrads, research scholars, faculty, HOD, director
   b) Apply finance: for Undergrads, research scholar
   c) Approve leave: for faculty, HOD, director, administrator
   d) Approve finance: for faculty, administrator
   e) Check leave status: for Undergrads, research scholars, faculty, HOD, director
   f) Check finance status: for Undergrads, research scholar
   g) Register
   h) Login

2. **Out-of-scope**:
   Performance related tests including efficiency and space-time complexities has not been performed.

3. **Items not tested**:
   The storing and retrieval of the objects during starting and shutting down of system is not done since file handling has not been implemented yet.

2 TEST PLAN

1. For each independent module unit testing is performed.

2. For modules from ‘a’ to ‘f’ Control flow testing is applied which includes complete branch and predicate coverage.

3. The modules involve procedural-call interface and shared-memory interface. So, attempts are made to perform appropriate system integration testing: both pairwise and end-to-end testing to minimize errors.

3 DETAILS OF UNIT TESTING

3.1 APPLY LEAVE: FOR UNDERGRADS

Function Name	apply_leave
Class	Undergrads
Input Parameters	**a (int)** – Number of leaves the user has applied for
	obj (Leave) – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if 'a' leaves can be granted or not ○ If yes, leave is applied and the application is forwarded to the Administrator for approval. ○ If not, an error message is given to user.
Output Parameters	None

Code Snippet

```
void Undergrads::apply_leave(int a, Leave obj)
{
   int b=obj.check_number();
   if(a<=b&&a>0)
   {
      cout<<"Leave applied"<<endl;
        obj.set_applied(a);
        obj.set_status("administrator");
        _administrator.add_leave(obj);
        Undergrads temp=_database.get_ug(obj.get_id());
        temp.set_leave_obj(obj);
        _database.addug(temp);
        Undergrads p=_database.get_ug(obj.get_id());
   }
   else
   {
      cout<<"Not applied,
      invalid application\n";
   }
}
```

3.1.1 Test items

The unit to be tested here is the function apply_leave (which facilitates the process of applying a leave), for an undergraduate student.

3.1.2 Features to be tested

→ Validity of the number of leaves applied .i.e. input parameter

→ Validity of the leave application .i.e. whether the requested number of leaves can be granted or not.

3.1.3 Item pass/ fail criteria

→ If 'a' is less than zero: the leave request is rejected by the system and a message is sent back to the applicant.

Rationale: Number of leaves applied for cannot be negative.

→ Once it is verified that the number of leaves applied for is positive, the system checks if 'a' leaves can be granted or not.

- If a > number of pending leaves, then the application is rejected with an appropriate error message to the applicant.
- If a <= number of pending leaves, then the application is accepted and forwarded to the administrator for approval.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.1.4 Test Cases

Test ID	Test Input	Local Variable values	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	b				
3.1.4.1	12	50	Leave applied	Leave applied	**PASS**	-
3.1.4.2	40	38	Not applied	Not applied	**PASS**	-
3.1.4.3	20	38	Leave applied	Leave applied	**PASS**	-
3.1.4.4	100	50	Not applied	Not applied	**PASS**	-
3.1.4.5	-20	50	Not applied	Not applied	**PASS**	-
3.1.4.6	-20	38	Not applied	Not applied	**PASS**	-

3.1.5 Test Result

The function apply_leave for Undergrads processed the inputs as expected for all the input cases.

3.2 APPLY LEAVE: FOR RESEARCH SCHOLAR

Function Name	apply_leave
Class	Research_scholar
Input Parameters	**a (int)** – Number of leaves the user has applied for
	obj (Leave) – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if 'a' leaves can be granted or not ○ If yes, leave is applied and the application is forwarded to the faculty (Research supervisor) for approval. ○ If not, an error message is given to user.
Output Parameters	None

Code Snippet

```
void Research_scholar::apply_leave(int a,Leave obj)
{
    int b=obj.check_number();
    if(a<=b&&a>0)
    {
        cout<<"Leave applied"<<endl;
        obj.set_applied(a);
        obj.set_status("faculty");
        Faculty f=_database.get_fac(faculty_id);
        f.add_leave(obj);
        _database.addfac(f);
        Research_scholar temp=_database.get_res(obj.get_id());
        temp.set_leave_obj(obj);
        _database.addres(temp);
    }
    else
    {
    cout<<"Not applied,invalid application\n";
    }
}
```

3.2.1 Test items

The unit to be tested here is the function apply_leave (which facilitates the process of applying a leave), for a research scholar.

3.2.2 Features to be tested

→ Validity of the number of leaves applied .i.e. input parameter

→ Validity of the leave application .i.e. whether the requested number of leaves can be granted or not.

3.2.3 Item pass/ fail criteria

→ If 'a' is less than zero: the leave request is rejected by the system and a message is sent back to the applicant.

Rationale: Number of leaves applied for cannot be negative.

→ Once it is verified that the number of leaves applied for is positive, the system checks if 'a' leaves can be granted or not.

- If a > number of pending leaves, then the application is rejected with an appropriate error message to the applicant.
- If a <= number of pending leaves, then the application is accepted and forwarded to the faculty (Research supervisor) for approval.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.2.4 Test Cases

Test ID	Test Input	Local Variable values	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	b				
3.2.4.1	12	50	Leave applied	Leave applied	**PASS**	-
3.2.4.2	40	38	Not applied	Not applied	**PASS**	-
3.2.4.3	20	38	Leave applied	Leave applied	**PASS**	-
3.2.4.4	100	50	Not applied	Not applied	**PASS**	-
3.2.4.5	-20	50	Not applied	Not applied	**PASS**	-
3.2.4.6	-20	38	Not applied	Not applied	**PASS**	-

3.2.5 Test Result

The function apply_leave for Research_scholar processed the inputs as expected for all the input cases.

3.3 APPLY LEAVE: FOR FACULTY

Function Name	apply_leave
Class	Faculty
Input Parameters	**a (int)** – Number of leaves the user has applied for
	obj (Leave) – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if 'a' leaves can be granted or not ○ If yes, leave is applied. ○ If the object obj is a Research scholar object, the application is forwarded to the Administrator for approval; else it is forwarded to the HOD of the concerned department for his approval. ○ If not, an error message is given to user.
Output Parameters	None

Code Snippet

```
void Faculty::apply_leave(int a,Leave obj)
{
    int b=obj.check_number();
    if(a<=b&&a>0)
    {
        obj.set_applied(a);
        obj.set_status("Hod");
        Hod h=_database.get_hod_dep(department);
        if(obj.get_designation()=="Research_scholar")
        {
            cout<<"Leave Forwarded"<<endl;
            obj.set_status("Administrator");
            Research_scholar temp=_database.get_res(obj.get_id());
            temp.set_leave_obj(obj);
            _database.addres(temp);
            _administrator.add_leave(obj);
        }
        else
        {
            cout<<"Leave applied"<<endl;
            Faculty temp=_database.get_fac(obj.get_id());
            temp.set_leave_obj(obj);
            _database.addfac(temp);
            h.add_leave(obj);
            _database.addhod(h);
        }
    }
    else
        cout<<"Not applied,invalid application\n";
}
```

3.3.1 Test items

The unit to be tested here is the function apply_leave (which facilitates the process of applying a leave), for a faculty.

3.3.2 Features to be tested

→ Validity of the number of leaves applied .i.e. input parameter
→ Validity of the leave application .i.e. whether the requested number of leaves can be granted or not.

3.3.3 Item pass/ fail criteria

→ If 'a' is less than zero: the leave request is rejected by the system and a message is sent back to the applicant.

Rationale: Number of leaves applied for cannot be negative.

→ Once it is verified that the number of leaves applied for is positive, the system checks if 'a' leaves can be granted or not.

- If a > number of pending leaves, then the application is rejected with an appropriate error message to the applicant.
- If a <= number of pending leaves and
  - obj is a Faculty object, then the application is accepted and forwarded to the HOD of the concerned department for approval.
  - obj is a Research_scholar object, then the application is accepted and forwarded to the administrator for approval. (This case occurs when a Faculty tries to approve the leave application of a research scholar under him.)

→ For the cases when 'a' is negative and 'a' exceeds the number of pending leaves, explicit testing for the situation when obj is a Research_scholar object; is NOT required.

Rationale: These 2 situations would already have been tested in the apply_leave function for Research_scholar when they try to apply for one.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.3.4 Test Cases

Test ID	Test Input		Local Variable values	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	obj	b				
3.3.4.1	12	Fac*	50	Leave applied (forwarded to HOD)	Leave applied (forwarded to HOD)	**PASS**	-
3.3.4.2	40	Fac*	38	Not applied	Not applied	**PASS**	-
3.3.4.3	20	Fac*	38	Leave applied (forwarded to HOD)	Leave applied (forwarded to HOD)	**PASS**	-
3.3.4.4	12	RS#	50	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.3.4.5	20	RS#	38	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.3.4.6	100	Fac*	50	Not applied	Not applied	**PASS**	-
3.3.4.7	-20	Fac*	50	Not applied	Not applied	**PASS**	-
3.3.4.8	-20	Fac*	38	Not applied	Not applied	**PASS**	-

(*) Fac : Faculty

(#) RS : Research Scholar

3.3.5 Test Result

The function apply_leave for Faculty processed the inputs as expected for all the input cases.

3.4 APPLY LEAVE: FOR HOD

Function Name	apply_leave
Class	HOD
Input Parameters	**a (int)** – Number of leaves the user has applied for
	obj (Leave) – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if 'a' leaves can be granted or not ○ If yes, leave is applied. ○ If the object obj is a Faculty object, the application is forwarded to the Director for his approval. ○ If obj is a HOD object it is again forwarded to the Director for his approval. ○ If not, an error message is given to user.
Output Parameters	None

Code Snippet

```
void  Hod::apply_leave(int a,Leave obj)
{
    int b=obj.check_number();
    if(a<=b&&a>0)
    {
        obj.set_applied(a);
        obj.set_status("Director");
        _director.add_leave(obj);
        if(obj.get_designation()=="Faculty")
        {
            cout<<"Leave forwarded"<<endl;
            Faculty temp=_database.get_fac(obj.get_id());
            temp.set_leave_obj(obj);
            _database.addfac(temp);
        }
        else
        {
            cout<<"Leave applied"<<endl;
            Hod temp=_database.get_hod(obj.get_id());
            temp.set_leave_obj(obj);
            _database.addhod(temp);
        }
    }
    else
    {
    cout<<"Not applied,invalid application\n";
    }
}
```

3.4.1 Test items

The unit to be tested here is the function apply_leave (which facilitates the process of applying a leave), for a HOD.

3.4.2 Features to be tested

→ Validity of the number of leaves applied .i.e. input parameter

→ Validity of the leave application .i.e. whether the requested number of leaves can be granted or not.

3.4.3 Item pass/ fail criteria

→ If 'a' is less than zero: the leave request is rejected by the system and a message is sent back to the applicant.

Rationale: Number of leaves applied for cannot be negative.

→ Once it is verified that the number of leaves applied for is positive, the system checks if 'a' leaves can be granted or not.

- If a > number of pending leaves, then the application is rejected with an appropriate error message to the applicant.
- If a <= number of pending leaves and
  - obj is a HOD object, then the application is accepted and forwarded to the Director for approval.
  - obj is a Faculty object, then the application is accepted and forwarded again to the Director for approval. (This case occurs when a HOD tries to approve the leave application of a faculty in his department.)

→ For the cases when 'a' is negative and 'a' exceeds the number of pending leaves, explicit testing for the situation when obj is a Faculty object; is NOT required.

Rationale: These 2 situations would already have been tested in the apply_leave function for Faculty when they try to apply for one.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.4.4 Test Cases

Test ID	Test Input		Local Variable values	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	obj	b				
3.4.4.1	12	HOD	50	Leave applied (forwarded to Director)	Leave applied (forwarded to Director)	**PASS**	-
3.4.4.2	40	HOD	38	Not applied	Not applied	**PASS**	-
3.4.4.3	20	HOD	38	Leave applied (forwarded to Director)	Leave applied (forwarded to Director)	**PASS**	-
3.4.4.4	12	Fac*	50	Leave applied (forwarded to Director)	Leave applied (forwarded to Director)	**PASS**	-
3.4.4.5	20	Fac*	38	Leave applied (forwarded to Director)	Leave applied (forwarded to Director)	**PASS**	-
3.4.4.6	100	HOD	50	Not applied	Not applied	**PASS**	-
3.4.4.7	-20	HOD	50	Not applied	Not applied	**PASS**	-
3.4.4.8	-20	HOD	38	Not applied	Not applied	**PASS**	-

(*) Fac : Faculty

3.4.5 Test Result

The function apply_leave for HOD processed the inputs as expected for all the input cases.

3.5 APPLY LEAVE: FOR DIRECTOR

Function Name	apply_leave
Class	Director
Input Parameters	**a (int)** – Number of leaves the user has applied for
	obj (Leave) – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if 'a' leaves can be granted or not ○ If yes, leave is applied. ○ If the object obj is a Director object, the application is forwarded to the Administrator for approval ○ If obj is a HOD or a Faculty object it is again forwarded to the Administrator for approval. ○ If not, an error message is given to user.
Output Parameters	None

3.5.1 Test items

The unit to be tested here is the function apply_leave (which facilitates the process of applying a leave), for the Director.

3.5.2 Features to be tested

→ Validity of the number of leaves applied .i.e. input parameter
→ Validity of the leave application .i.e. whether the requested number of leaves can be granted or not.

3.5.3 Item pass/ fail criteria

→ If 'a' is less than zero: the leave request is rejected by the system and a message is sent back to the applicant.

Rationale: Number of leaves applied for cannot be negative.

→ Once it is verified that the number of leaves applied for is positive, the system checks if 'a' leaves can be granted or not.

- If a > number of pending leaves, then the application is rejected with an appropriate error message to the applicant.
- If a <= number of pending leaves and
  - obj is a HOD object or a Faculty object, then the application is accepted and forwarded to the Administrator for approval. (This case occurs when the Director tries to approve the leave application of a HOD or Faculty.)

- obj is a Director object, then the application is accepted and forwarded again to the Administrator for approval.

→ For the cases when ‘a’ is negative and ‘a’ exceeds the number of pending leaves, explicit testing for the situation when obj is a HOD/Faculty object; is NOT required.

Rationale: These 2 situations would already have been tested in the apply_leave function for HOD/Faculty when they try to apply for one.

→ If the system abides by these criteria, the test is deemed to have been passed.

Code Snippet

```
void Director::apply_leave(int a,Leave obj)
{
    int b=obj.check_number();
    if(a<=b&&a>0)
    {
        obj.set_applied(a);
        obj.set_status("administrator");
        _administrator.add_leave(obj);
        if(obj.get_designation()=="Faculty")
        {
            cout<<"Leave forwarded"<<endl;
            Faculty temp=_database.get_fac(obj.get_id());
            temp.set_leave_obj(obj);
            _database.addfac(temp);
        }
        else if(obj.get_designation()=="Hod")
        {
            cout<<"Leave forwarded"<<endl;
            Hod temp=_database.get_hod(obj.get_id());
            temp.set_leave_obj(obj);
            _database.addhod(temp);
        }
        else
        {
            cout<<"Leave applied"<<endl;
            Director temp=_database.get_dir(obj.get_id());
            temp.set_leave_obj(obj);
            _database.adddir(temp);
        }
    }
    else
    {
        cout<<"Not applied, invalid application\n";
    }
}
```

3.5.4 Test Cases

Test ID	Test Input		Local Variable values	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	obj	b				
3.5.4.1	12	Director	50	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.5.4.2	40	Director	38	Not applied	Not applied	**PASS**	-
3.5.4.3	20	Director	38	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.5.4.4	12	HOD	50	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.5.4.5	20	HOD	38	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.5.4.6	12	Fac*	50	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.5.4.7	20	Fac*	38	Leave applied (forwarded to Administrator)	Leave applied (forwarded to Administrator)	**PASS**	-
3.5.4.8	100	Director	50	Not applied	Not applied	**PASS**	-
3.5.4.9	-20	Director	50	Not applied	Not applied	**PASS**	-
3.5.4.10	-20	Director	38	Not applied	Not applied	**PASS**	-

(*) Fac : Faculty

3.5.5 Test Result

The function apply_leave for the Director processed the inputs as expected for all the input cases.

3.6 APPLY FINANCE: FOR UNDERGRADS

Function Name	apply_finance
Class	Undergrads
Input Parameters	**a (double)** – The financial assistance the user has applied for
	obj (Finance) – Finance object with details of financial income, amount received currently, financial application status, etc
Processing	○ Checks if 'a' amount can be granted or not (amount 'a' can be granted if the total assistance does not exceed 2 lakhs) ○ If yes, financial assistance is applied and the application is forwarded to the Administrator for approval. ○ If not, an error message is given to user.
Output Parameters	None

Code Snippet

```
void Undergrads::apply_finance(double a,Finance obj)
{
    double b=obj.check_amount();
    if(a+b<=200000&&a>0)
    {
        cout<<"Financial application applied"<<endl;
        obj.set_applied(a);
        obj.set_status("administrator");
        _administrator.add_finance(obj);
        Undergrads temp=_database.get_ug(obj.get_id());
        temp.set_finance_obj(obj);
        _database.addug(temp);
    }
    else
    {
        cout<<"Not applied,invalid application\n";
    }
}
```

3.6.1 Test items

The unit to be tested here is the function apply_finance (which facilitates the process of applying for financial assistance), for an undergraduate student.

3.6.2 Features to be tested

→ Validity of the amount applied for .i.e. input parameter

→ Validity of the application .i.e. whether the amount requested can be granted or not.

3.6.3 Item pass/ fail criteria

→ If 'a' is less than zero: the request is rejected by the system and a message is sent back to the applicant.

Rationale: Amount applied for cannot be negative.

→ Once it is verified that the amount applied for is positive, the system checks if the amount 'a' can be granted or not.

- If a + Amount received now as financial assistance > 2,00,000 (OR) family income exceeds 5 lakhs, then the application is rejected with an appropriate error message to the applicant.
- If a + Amount received now as financial assistance <= 2,00,000 (AND) family income is below 5 lakhs, then the application is accepted and forwarded to the Administrator for approval.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.6.4 Test Cases

Test ID	Test Input	Local Variables	Family Income	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	b					
3.6.4.1	1,00,000	0	2,00,000	Applied	Applied	**PASS**	-
3.6.4.2	1,50,000	1,00,000	2,00,000	Not applied	Not applied	**PASS**	-
3.6.4.3	50,000	1,00,000	2,00,000	Applied	Applied	**PASS**	-
3.6.4.4	10,00,000	0	2,00,000	Not applied	Not applied	**PASS**	-
3.6.4.5	-20,000	0	2,00,000	Not applied	Not applied	**PASS**	-
3.6.4.6	-20,000	1,00,000	2,00,000	Not applied	Not applied	**PASS**	-
3.6.4.7	1,00,000	0	8,00,000	Not applied	Not applied	**PASS**	-
3.6.4.8	1,50,000	1,00,000	8,00,000	Not applied	Not applied	**PASS**	-
3.6.4.9	50,000	1,00,000	8,00,000	Not applied	Not applied	**PASS**	-
3.6.4.10	10,00,000	0	8,00,000	Not applied	Not applied	**PASS**	-
3.6.4.11	-20,000	0	8,00,000	Not applied	Not applied	**PASS**	-
3.6.4.12	-20,000	1,00,000	8,00,000	Not applied	Not applied	**PASS**	-

3.6.5 Test Result

The function apply_finance for Undergrads processed the inputs as expected for all the input cases.

3.7 APPLY FINANCE: FOR RESEARCH SCHOLARS

Function Name	apply_finance
Class	Research_scholar
Input Parameters	**a (double)** – The financial assistance the user has applied for
	obj (Finance) –Finance object with details of financial income, amount received currently, financial application status, etc
Processing	○ Checks if 'a' amount can be granted or not (amount 'a' can be granted if the total assistance does not exceed 2 lakhs) ○ If yes, financial assistance is applied and the application is forwarded to the faculty (Research supervisor) for approval. ○ If not, an error message is given to user.
Output Parameters	None

Code Snippet

```
void Research_scholar::apply_finance(double a,Finance obj)
{
    double b=obj.check_amount();
    if(a+b<=200000&&a>0)
    {
        cout<<"Financial application applied"<<endl;
        obj.set_applied(a);
        obj.set_status("faculty");
        Faculty f=_database.get_fac(faculty_id);
        f.add_finance(obj);
        _database.addfac(f);
        Research_scholar temp=_database.get_res(obj.get_id());
        temp.set_finance_obj(obj);
    }
    else
    {
        cout<<"Not applied,invalid application\n";
    }
}
```

3.7.1 Test items

The unit to be tested here is the function apply_finance (which facilitates the process of applying for financial assistance), for a research scholar.

3.7.2 Features to be tested

→ Validity of the amount applied for .i.e. input parameter

→ Validity of the application .i.e. whether the amount requested can be granted or not.

3.7.3 Item pass/ fail criteria

→ If 'a' is less than zero: the request is rejected by the system and a message is sent back to the applicant.

Rationale: Amount applied for cannot be negative.

→ Once it is verified that the amount applied for is positive, the system checks if the amount 'a' can be granted or not.

- If a + Amount received now as financial assistance > 2,00,000 (OR) family income exceeds 5 lakhs, then the application is rejected with an appropriate error message to the applicant.
- If a + Amount received now as financial assistance <= 2,00,000 (AND) family income is below 5 lakhs, then the application is accepted and forwarded to the faculty (Research supervisor) for approval.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.7.4 Test Cases

Test ID	Test Input	Local Variables	Family Income	Expected Output	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
	a	b					
3.7.4.1	1,00,000	0	2,00,000	Applied	Applied	**PASS**	-
3.7.4.2	1,50,000	1,00,000	2,00,000	Not applied	Not applied	**PASS**	-
3.7.4.3	50,000	1,00,000	2,00,000	Applied	Applied	**PASS**	-
3.7.4.4	10,00,000	0	2,00,000	Not applied	Not applied	**PASS**	-
3.7.4.5	-20,000	0	2,00,000	Not applied	Not applied	**PASS**	-
3.7.4.6	-20,000	1,00,000	2,00,000	Not applied	Not applied	**PASS**	-
3.7.4.7	1,00,000	0	8,00,000	Not applied	Not applied	**PASS**	-
3.7.4.8	1,50,000	1,00,000	8,00,000	Not applied	Not applied	**PASS**	-
3.7.4.9	50,000	1,00,000	8,00,000	Not applied	Not applied	**PASS**	-
3.7.4.10	10,00,000	0	8,00,000	Not applied	Not applied	**PASS**	-
3.7.4.11	-20,000	0	8,00,000	Not applied	Not applied	**PASS**	-
3.7.4.12	-20,000	1,00,000	8,00,000	Not applied	Not applied	**PASS**	-

3.7.5 Test Result

The function apply_finance for Research_scholar processed the inputs as expected for all the input cases.

3.8 APPROVE LEAVE: FOR FACULTY

Function Name	approve_leave
Class	Faculty
Input Parameters	**obj (Leave)** – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if the leave application corresponding to obj is pending for approval or not. ○ If the leave application is pending for approval, then the control is handed over to the apply_leave function (Faculty) using the object obj as the parameter. (Once the control is in the apply_leave function, the path corresponding to a Research_scholar is executed.) ○ The leave application corresponding to object obj is removed from the list of applications pending for the faculty's approval.
Output Parameters	None

Code Snippet

```
void Faculty::approve_leave(Leave obj)
{
    if(!find_leave(obj))
        throw exception();
    apply_leave(obj.get_applied(),obj);
    remove_leave(obj);
    Faculty f=(*this);
    _database.addfac(f);
}
```

3.8.1 Test items

The unit to be tested here is the function approve_leave (which facilitates the process of approving a leave), for a faculty.

3.8.2 Features to be tested

→ The primary feature to be tested for this function is the presence or absence of outstanding leave applications

3.8.3 Item pass/ fail criteria

→ If the leave application corresponding to obj is not pending for approval, an appropriate message is given to the user.

→ If the leave application corresponding to object obj is pending for the Faculty's approval, then:

- apply_leave function (for Faculty) is invoked with obj (Leave object for Research scholar) as its parameter.
- The research scholar's application is processed and forwarded to the Administrator for approval.
- The list of pending leave applications is updated by removing this application from it.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.8.4 Test Cases

Test ID		User input *(Explicitly provided)*		Input Parameter *(Implicitly computed)*	Expected Output	Actual Output	Result (Pass/ Fail)	Severity of the failure (if any)
		user_id	**designation**	**obj**				
3.8.4.1	#	Nil	Nil	Nil	No pending applications	No pending applications	**PASS**	-
3.8.4.2		5	Research_ scholar	Research_ scholar	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
3.8.4.3		5	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.8.4.4		7 (Wrong ID)	Research_ scholar	-	Not found	Not found	**PASS**	-
3.8.4.5		7 (Wrong ID)	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.8.4.6		3	Research_ scholar	Research_ scholar	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
		4	Research_ scholar	Research_ scholar	Leave approved and forwarded	Leave approved and forwarded		

- The list of outstanding applications used for unit testing; described in the table that follows

# - Pending Leave Applications				
Test ID	**User ID**	**Applied number of leaves**	**Designation**	**Comments**
3.8.4.1	Nil	Nil	Nil	No Pending Applications
3.8.4.2	5	12	Research_scholar	1 pending application
3.8.4.3	5	12	Research_scholar	1 pending application
3.8.4.4	5	12	Research_scholar	1 pending application
3.8.4.5	5	12	Research_scholar	1 pending application
3.8.4.6	3	15	Research_scholar	2 pending applications
	4	18	Research_scholar	

3.8.5 Test Result

The function approve_leave for Faculty processed the inputs as expected for all the input cases.

3.9 APPROVE LEAVE: FOR HOD

Function Name	approve_leave
Class	HOD
Input Parameters	**obj (Leave)** – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if the leave application corresponding to obj is pending for approval or not. ○ If the leave application is pending for approval, then the control is handed over to the apply_leave function (HOD) using the object obj as the parameter. (Once the control is in the apply_leave function, the path corresponding to a Faculty is executed.) ○ The leave application corresponding to object obj is removed from the list of applications pending for the HOD's approval.
Output Parameters	None

Code Snippet

```
void Hod::approve_leave(Leave obj)
{
    if(!find_leave(obj))
        throw exception();
    apply_leave(obj.get_applied(),obj);
    remove_leave(obj);
    Hod f=(*this);
    _database.addhod(f);

}
```

3.9.1 Test items

The unit to be tested here is the function approve_leave (which facilitates the process of approving a leave), for a department HOD.

3.9.2 Features to be tested

→ The primary feature to be tested for this function is the presence or absence of outstanding leave applications

3.9.3 Item pass/ fail criteria

→ If the leave application corresponding to obj is not pending for approval, an appropriate message is given to the user.

→ If the leave application corresponding to object obj is pending for the HOD's approval, then:

- apply_leave function (for HOD) is invoked with obj (leave object for Faculty) as its parameter.
- The Faculty's application is processed and forwarded to the Director for approval.
- The list of pending leave applications is updated by removing this application from it.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.9.4 Test Cases

Test ID		User input *(Explicitly provided)*		Input Parameter *(Implicitly computed)*	Expected Output	Actual Output	Result (Pass/ Fail)	Severity of the failure (if any)
		user_id	**designation**	**obj**				
3.9.4.1	#	Nil	Nil	Nil	No pending applications	No pending applications	**PASS**	-
3.9.4.2		5	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
3.9.4.3		5	Director (Wrong designation)	-	Not found	Not found	**PASS**	-
3.9.4.4		7 (Wrong ID)	Faculty	-	Not found	Not found	**PASS**	-
3.9.4.5		7 (Wrong ID)	Director (Wrong designation)	-	Not found	Not found	**PASS**	-
3.9.4.6		3	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
		4	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded		

- The list of outstanding applications used for unit testing; described in the table that follows

# - Pending Leave Applications				
Test ID	**User ID**	**Applied number of leaves**	**Designation**	**Comments**
3.9.4.1	Nil	Nil	Nil	No Pending Applications
3.9.4.2	5	12	Faculty	1 pending application
3.9.4.3	5	12	Faculty	1 pending application
3.9.4.4	5	12	Faculty	1 pending application
3.9.4.5	5	12	Faculty	1 pending application
3.9.4.6	3	15	Faculty	2 pending applications
	4	18	Faculty	

3.9.5 Test Result

The function approve_leave for HOD processed the inputs as expected for all the input cases.

3.10 APPROVE LEAVE: FOR DIRECTOR

Function Name	approve_leave
Class	Director
Input Parameters	**obj (Leave)** – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if the leave application corresponding to obj is pending for approval or not. ○ If the leave application is pending for approval, then the control is handed over to the apply_leave function (HOD or Faculty) using the object obj as the parameter. (Once the control is in the apply_leave function, the path corresponding to the HOD or Faculty is executed.) ○ The leave application corresponding to object obj is removed from the list of applications pending for the Director's approval.
Output Parameters	None

Code Snippet

```
void Director::approve_leave(Leave obj)
{
    if(!find_leave(obj))
        throw exception();
    apply_leave(obj.get_applied(),obj);
    remove_leave(obj);
    Director d=(*this);
    _database.adddir(d);
}
```

3.10.1 Test items

The unit to be tested here is the function approve_leave (which facilitates the process of approving a leave), for the Director.

3.10.2 Features to be tested

→ The primary feature to be tested for this function is the presence or absence of outstanding leave applications

3.10.3 Item pass/ fail criteria

→ If the leave application corresponding to obj is not pending for approval, an appropriate message is given to the user.

→ If the leave application corresponding to object obj is pending for the Director's approval, then:

- apply_leave function (for the Director) is invoked with obj (leave object for HOD or Faculty) as its parameter.
- The HOD's or Faculty's application is processed and forwarded to the Administrator for approval.
- The list of pending leave applications is updated by removing this application from it.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.10.4 Test Cases

Test ID	#	User input *(Explicitly provided)*		Input Parameter *(Implicitly computed)*	Expected Output	Actual Output	Result (Pass/ Fail)	Severity of the failure (if any)
		user_id	**designation**	**obj**				
3.10.4.1		Nil	Nil	Nil	No pending applications	No pending applications	**PASS**	-
3.10.4.2		5	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
3.10.4.3		5	Director (Wrong designation)	-	Not found	Not found	**PASS**	-
3.10.4.4		7 (Wrong ID)	Faculty	-	Not found	Not found	**PASS**	-
3.10.4.5		7 (Wrong ID)	Director (Wrong designation)	-	Not found	Not found	**PASS**	-
3.10.4.6		3	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
		4	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded		
3.10.4.7		1	HOD	HOD	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-

3.10.4.8		1	Director (Wrong designation)	-	Not found	Not found	**PASS**	-
3.10.4.9		9 (Wrong ID)	HOD	-	Not found	Not found	**PASS**	-
3.10.4.10		7 (Wrong ID)	Director (Wrong designation)	-	Not found	Not found	**PASS**	-
3.10.4.11		1	HOD	HOD	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
		2	HOD	HOD	Leave approved and forwarded	Leave approved and forwarded		
3.10.4.12		3	Faculty	Faculty	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-
		1	HOD	HOD	Leave approved and forwarded	Leave approved and forwarded	**PASS**	-

- The list of outstanding applications used for unit testing; described in the table that follows

# - Pending Leave Applications				
Test ID	**User ID**	**Applied number of leaves**	**Designation**	**Comments**
3.10.4.1	Nil	Nil	Nil	No Pending Applications
3.10.4.2	5	12	Faculty@	1 pending application
3.10.4.3	5	12	Faculty@	1 pending application
3.10.4.4	5	12	Faculty@	1 pending application
3.10.4.5	5	12	Faculty@	1 pending application
3.10.4.6	3	15	Faculty@	2 pending applications
	4	18	Faculty@	

3.10.4.7	1	20	HOD	1 pending application
3.10.4.8	1	20	HOD	1 pending application
3.10.4.9	1	20	HOD	1 pending application
3.10.4.10	1	20	HOD	1 pending application
3.10.4.11	1	20	HOD	2 pending applications
	2	25	HOD	
3.10.4.12	3	15	Faculty[@]	2 pending applications
	1	20	HOD	

@ - After approval by the corresponding department HOD

3.10.5 Test Result

The function approve_leave for the Director processed the inputs as expected for all the input cases.

3.11 APPROVE LEAVE: FOR ADMINISTRATOR

Function Name	approve_leave
Class	Administrator
Input Parameters	**obj (Leave)** – Leave object with details of designation, number of pending leaves, leave application status, etc
Processing	○ Checks if the leave application corresponding to obj is pending for approval or not. ○ If the leave application corresponding to obj (Undergrad/ Research scholar/ Faculty/ HOD/ Director) is pending for approval, then the application is approved. ○ The leave application corresponding to object obj is removed from the list of applications pending for the Administrator's approval. ○ Set the field corresponding to the number of applied leaves, in obj to zero. ○ Corresponding changes are made in the database as well.
Output Parameters	None

3.11.1 Test items

The unit to be tested here is the function approve_leave (which facilitates the process of approving a leave), for the Administrator.

3.11.2 Features to be tested

→ The primary feature to be tested for this function is the presence or absence of outstanding leave applications

3.11.3 Item pass/ fail criteria

→ If the leave application corresponding to obj is not pending for approval, an appropriate message is given to the user.

→ If the leave application corresponding to object obj is pending for the Administrator's approval, then:

- The application is approved.
- The applied leaves (awaiting approval) field for the object is set to zero.
- The list of pending leave applications is updated by removing this application from it.

→ If the system abides by these criteria, the test is deemed to have been passed.

Code Snippet:

```
void Administrator::approve_leave(Leave obj)
{
    if(!find_leave(obj))
        throw exception();
    obj.set_status("approved");
    obj.deduct(obj.get_applied());
    obj.set_applied(0);
    remove_leave(obj);
    if(obj.get_designation()=="Undergrads")
    {
        Undergrads temp=_database.get_ug(obj.get_id());
        temp.set_leave_obj(obj);
        _database.addug(temp);
    }
    else if(obj.get_designation()=="Research_scholar")
    {
        Research_scholar temp=_database.get_res(obj.get_id());
        temp.set_leave_obj(obj);
        _database.addres(temp);
    }
    else if(obj.get_designation()=="Faculty")
    {
        Faculty temp=_database.get_fac(obj.get_id());
        temp.set_leave_obj(obj);
        _database.addfac(temp);
    }
    else if(obj.get_designation()=="Hod")
    {
        Hod temp=_database.get_hod(obj.get_id());
        temp.set_leave_obj(obj);
        _database.addhod(temp);
    }
    else
    {
        Director temp=_database.get_dir(obj.get_id());
        temp.set_leave_obj(obj);
        _database.adddir(temp);
    }
    cout<<"Leave approved"<<endl;
    _administrator=*this;
}
```

3.11.4 Test Cases

Test ID	#	User input *(Explicitly provided)*		Input Parameter *(Implicitly computed)*	Expected Output	Actual Output	Result (Pass/ Fail)	Severity of the failure (if any)
		user_id	**designation**	**obj**				
3.11.4.1		Nil	Nil	Nil	No applications pending for approval	No applications pending for approval	**PASS**	-
3.11.4.2		123	Director	Director	Leave approved	Leave approved	**PASS**	-
3.11.4.3		123	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.11.4.4		1	HOD	HOD	Leave approved	Leave approved	**PASS**	-
3.11.4.5		7 (Wrong ID)	HOD	-	Not found	Not found	**PASS**	-
3.11.4.6		2	Faculty	Faculty	Leave approved	Leave approved	**PASS**	-
3.11.4.7		2	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.11.4.8		3	Research_ scholar	Research_ scholar	Leave approved	Leave approved	**PASS**	-
3.11.4.9		7 (Wrong ID)	Research_ scholar	-	Not found	Not found	**PASS**	-
3.11.4.10		4	Undergrads	Undergrads	Leave approved	Leave approved	**PASS**	-
3.11.4.11		7 (Wrong ID)	Undergrads	-	Not found	Not found	**PASS**	-
3.11.4.12		123	Director	Director	Leave approved	Leave approved	**PASS**	-
		1	HOD	HOD	Leave approved	Leave approved		

		2	Faculty	Faculty	Leave approved	Leave approved		
		3	Research_ scholar	Research_ scholar	Leave approved	Leave approved		
		4	Undergrads	Undergrads	Leave approved	Leave approved		

- The list of outstanding applications used for unit testing; described in the table that follows

# - Pending Leave Applications				
Test ID	**User ID**	**Applied number of leaves**	**Designation**	**Comments**
3.11.4.1	Nil	Nil	Nil	No Pending Applications
3.11.4.2	123	25	Director	1 pending application
3.11.4.3	123	25	Director	1 pending application
3.11.4.4	1	21	HOD [a]	1 pending application
3.11.4.5	1	21	HOD [a]	1 pending application
3.11.4.6	2	18	Faculty [b]	1 pending application
3.11.4.7	2	18	Faculty [b]	1 pending application
3.11.4.8	3	15	Research_scholar [c]	1 pending application
3.11.4.9	3	15	Research_scholar [c]	1 pending application
3.11.4.10	4	12	Undergrad [d]	1 pending application
3.11.4.11	4	12	Undergrad [d]	1 pending application
3.11.4.12	123	25	Director	5 pending applications
	1	21	HOD	
	2	18	Faculty	
	3	15	Research_scholar	
	4	12	Undergrad	

Legend:

- HOD [a], Faculty [b], Research_scholar [c], Undergrad [d] – Applications that have reached the Administrator after approval by the concerned authorities in the defined hierarchy chain
- There are several other possibilities for the list of pending leave applications. Further for each of these cases, there are 2 possible situations depending on the user inputs:
  - entered user_id and designation are correct and matching.
  - any one or both of the user_id and designation are incorrect.

  The application is approved if and only if both the inputs are correct.

  Here 12 of those cases are shown to depict a few.

3.11.5 Test Result

The function approve_leave for the Administrator processed the inputs as expected for the selected input cases.

3.12 APPROVE FINANCE: FOR FACULTY

Function Name	approve_finance
Class	Faculty
Input Parameters	**obj (Finance)** – Finance object with details of financial income, amount received currently, financial application status, etc
Processing	○ Checks if the application corresponding to obj is pending for approval or not. ○ If the finance application from a research scholar is pending for approval, the application is approved and forwarded to the Administrator for approval. ○ The leave application corresponding to object obj is removed from the list of applications pending for the faculty's approval.
Output Parameters	None

Code Snippet

```
void Faculty::approve_finance(Finance obj)
{
    if(!find_finance(obj))throw exception();
    obj.set_status("Administrator");
    remove_finance(obj);
    Faculty f=(*this);
    _database.addfac(f);
    if(obj.get_designation()=="Research_scholar")
    {
        Research_scholar temp=_database.get_res(obj.get_id());
        temp.set_finance_obj(obj);
        _database.addres(temp);
        _administrator.add_finance(obj);
        cout<<"Financial application forwarded"<<endl;
    }
}
```

3.12.1 Test items

The unit to be tested here is the function approve_finance (which facilitates the process of approving a financial application), for a faculty.

3.12.2 Features to be tested

→ The primary feature to be tested for this function is the presence or absence of outstanding financial applications.

3.12.3 Item pass/ fail criteria

→ If the application corresponding to obj is not pending for approval, an appropriate message is given to the user.

→ If the leave application corresponding to object obj is pending for the Faculty's approval, then:

- The research scholar's application is processed and forwarded to the Administrator for approval.
- The list of pending financial applications is updated by removing this application from it.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.12.4 Test Cases

Test ID		User input *(Explicitly provided)*		Input Parameter *(Implicitly computed)*	Expected Output	Actual Output	Result (Pass/ Fail)	Severity of the failure (if any)
		user_id	**designation**	**obj**				
3.12.4.1	#	Nil	Nil	Nil	No pending applications	No pending applications	**PASS**	-
3.12.4.2		5	Research_ scholar	Research_ scholar	Approved and forwarded	Approved and forwarded	**PASS**	-
3.12.4.3		5	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.12.4.4		7 (Wrong ID)	Research_ scholar	-	Not found	Not found	**PASS**	-
3.12.4.5		7 (Wrong ID)	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.12.4.6		3	Research_ scholar	Research_ scholar	Approved and forwarded	Approved and forwarded	**PASS**	-
		4	Research_ scholar	Research_ scholar	Approved and forwarded	Approved and forwarded		

- The list of outstanding applications used for unit testing; described in the table that follows

# - Pending Financial Assistance Applications				
Test ID	**User ID**	**Applied amount**	**Designation**	**Comments**
3.12.4.1	Nil	Nil	Nil	No Pending Applications
3.12.4.2	5	1,00,000	Research_scholar	1 pending application
3.12.4.3	5	1,00,000	Research_scholar	1 pending application
3.12.4.4	5	1,00,000	Research_scholar	1 pending application
3.12.4.5	5	1,00,000	Research_scholar	1 pending application
3.12.4.6	3	1,50,000	Research_scholar	2 pending applications
	4	1,80,000	Research_scholar	

3.12.5 Test Result

The function approve_finance for Faculty processed the inputs as expected for all the input cases.

3.13 APPROVE FINANCE: FOR ADMINISTRATOR

Function Name	approve_leave
Class	Administrator
Input Parameters	**obj (Finance)** – Finance object with details of financial income, amount received currently, financial application status, etc
Processing	○ Checks if the application corresponding to obj is pending for approval or not. ○ If the finance application from an undergrad or research scholar is pending for approval, the application is approved and the corresponding changes are made. ○ The leave application corresponding to object obj is removed from the list of applications pending for the Administrator's approval.
Output Parameters	None

Code Snippet

```
void Administrator::approve_finance(Finance obj)
{
    if(!find_finance(obj))
        throw exception();
    obj.set_status("approved");
    obj.add(obj.get_applied());
    obj.set_applied(0);
    remove_finance(obj);
    if(obj.get_designation()=="Undergrads")
    {
        Undergrads temp=_database.get_ug(obj.get_id());
        temp.set_finance_obj(obj);
        _database.addug(temp);
    }
    else if(obj.get_designation()=="Research_scholar")
    {
        Research_scholar temp=_database.get_res(obj.get_id());
        temp.set_finance_obj(obj);
        _database.addres(temp);
    }
    _administrator=(*this);
    cout<<"FInancial application approved"<<endl;
}
```

3.13.1 Test items

The unit to be tested here is the function approve_finance (which facilitates the process of approving a financial application), for the Administrator.

3.13.2 Features to be tested

→ The primary feature to be tested for this function is the presence or absence of outstanding financial applications.

3.13.3 Item pass/ fail criteria

→ If the application corresponding to obj is not pending for approval, an appropriate message is given to the user.

→ If the leave application corresponding to object obj is pending for the Administrator's approval, then:

- The Undergrad's or research scholar's application is approved.
- The list of pending financial applications is updated by removing this application from it.

→ If the system abides by these criteria, the test is deemed to have been passed.

3.13.4 Test Cases

Test ID		**User input** *(Explicitly provided)*		**Input Parameter** *(Implicitly computed)*	**Expected Output**	**Actual Output**	**Result (Pass/ Fail)**	**Severity of the failure (if any)**
		user_id	**designation**	**obj**				
3.13.4.1	#	Nil	Nil	Nil	No pending applications	No pending applications	**PASS**	-
3.13.4.2		5	Research_ scholar	Research_ scholar	Approved	Approved	**PASS**	-
3.13.4.3		5	HOD (Wrong designation)	-	Not found	Not found	**PASS**	-
3.13.4.4		4	Undergrads	Undergrads	Approved	Approved	**PASS**	-
3.13.4.5		7 (Wrong ID)	Undergrads	-	Not found	Not found	**PASS**	-
3.13.4.6		1	Undergrads	Undergrads	Approved	Approved	**PASS**	-
		2	Research_ scholar	Research_ scholar	Approved	Approved		

- The list of outstanding applications used for unit testing; described in the table that follows

# - Pending Financial Assistance Applications				
Test ID	**User ID**	**Applied amount**	**Designation**	**Comments**
3.13.4.1	Nil	Nil	Nil	No Pending Applications
3.13.4.2	5	1,00,000	Research_scholar	1 pending application
3.13.4.3	5	1,00,000	Research_scholar	1 pending application
3.13.4.4	4	1,50,000	Undergrads	1 pending application
3.13.4.5	4	1,50,000	Undergrads	1 pending application
3.13.4.6	1	1,25,000	Undergrads	2 pending applications
	2	1,80,000	Research_scholar	

3.13.5 Test Result

The function approve_finance for Administrator processed the inputs as expected for all the input cases.

.

3.14 CHECK LEAVE STATUS

Function Name	get_status
Class	Leave
Invoked for	**obj (Leave)** – Leave object with details of designation, number of pending leaves, leave application status, etc
Input Parameters	None
Processing	None (since it is defined to be just a getter function)
Output Parameters	Return the string stored in the status field of the Leave object

Code Snippet

```
string get_status()
{
    return status;
}
```

3.14.1 Test items

The unit to be tested here is the function get_status (which facilitates the process of getting a leave application status), for all the non-administrative users.
This function is placed in the class Leave and hence can be used to display the status for any user using the system.

3.14.2 Features to be tested

The only feature to be tested for this function is the status it returns

3.14.3 Item pass/ fail criteria

If the correct status is displayed when invoked for a particular object, the test is deemed to have been passed.

3.14.4 Test cases

Test ID	Invoked for	Actual Application Status	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
3.14.4.1	Undergrads	Not applied	Not applied	**PASS**	-
3.14.4.2	Undergrads	Administrator	Administrator	**PASS**	-
3.14.4.3	Undergrads	Approved	Approved	**PASS**	-
3.14.4.4	Research_scholar	Not applied	Not applied	**PASS**	-
3.14.4.5	Research_scholar	Faculty	Faculty	**PASS**	-
3.14.4.6	Research_scholar	Administrator	Administrator	**PASS**	-
3.14.4.7	Research_scholar	Approved	Approved	**PASS**	-
3.14.4.8	Faculty	Not applied	Not applied	**PASS**	-
3.14.4.9	Faculty	HOD	HOD	**PASS**	-
3.14.4.10	Faculty	Director	Director	**PASS**	-
3.14.4.11	Faculty	Administrator	Administrator	**PASS**	-
3.14.4.12	Faculty	Approved	Approved	**PASS**	-
3.14.4.13	HOD	Not applied	Not applied	**PASS**	-
3.14.4.14	HOD	Director	Director	**PASS**	-
3.14.4.15	HOD	Administrator	Administrator	**PASS**	-
3.14.4.16	HOD	Approved	Approved	**PASS**	-
3.14.4.17	Director	Not applied	Not applied	**PASS**	-
3.14.4.18	Director	Administrator	Administrator	**PASS**	-
3.14.4.19	Director	Approved	Approved	**PASS**	-

3.14.5 Test Result

The function get_status produced the output as expected for all the test cases

3.15 CHECK FINANCE STATUS

Function Name	get_status
Class	Finance
Invoked for	**obj (Finance)** – Finance object with details of financial income, amount received currently, financial application status, etc
Input Parameters	None
Processing	None (since it is defined to be just a getter function)
Output Parameters	Return the string stored in the status field of the Finance object

Code Snippet

```
string get_status()
{
    return status;
}
```

3.15.1 Test items

The unit to be tested here is the function get_status (which facilitates the process of getting a financial application status), for the Undergrads and Research scholars. This function is placed in the class Leave and hence can be used to display the status for both Undergrads and research scholars.

3.15.2 Features to be tested

The only feature to be tested for this function is the status it returns

3.15.3 Item pass/ fail criteria

If the correct status is displayed when invoked for a particular object, the test is deemed to have been passed.

3.14.4 Test cases

Test ID	Invoked for	Actual Application Status	Actual Output	Result (Pass/Fail)	Severity of the failure (if any)
3.14.4.1	Undergrads	Not applied	Not applied	**PASS**	-
3.14.4.2	Undergrads	Administrator	Administrator	**PASS**	-
3.14.4.3	Undergrads	Approved	Approved	**PASS**	-
3.14.4.4	Research_scholar	Not applied	Not applied	**PASS**	-
3.14.4.5	Research_scholar	Faculty	Faculty	**PASS**	-
3.14.4.6	Research_scholar	Administrator	Administrator	**PASS**	-
3.14.4.7	Research_scholar	Approved	Approved	**PASS**	-

3.14.5 Test Result

The function get_status produced the output as expected for all the test cases

3.15 REGISTER

Function Name	Register
Class	Utility_class
Input Parameters to the function	None
User inputs	Choice of designation (from a menu)
	Name
	ID
Processing	○ Checks if ID is already in use by another user of the same designation. ○ If ID is in use, registration with the inputted ID is not permitted. ○ Further, the system allows registration only if the higher authorities exist in the database. Eg: A faculty of a department (say CSE) can register himself only if the CSE HOD has registered himself in the database. ○ For an Undergrad user, family income is accepted during registration ○ For an Research_scholar, family income and research supervisor's ID are accepted during registration
Output Parameters	None

3.15.1 Test items

The unit to be tested here is the function Register (which facilitates the process of registration).

3.15.2 Features to be tested

- Uniqueness of ID
- Hierarchical registration

3.15.3 Item pass/ fail criteria

- The system should not allow more than one user of a particular designation to have the same ID
- Registration should be allowed only if the higher authorities exist in the database

Code Snippet

```
void Utility_class::Register()
{
    string name,desig;
    int opt;
    double id;
    cout<<"Register as\n"<<"1.Undergrads\n"<<"2.Research scholar\n"<<
    "3.Faculty\n"<<"4.Hod\n"<<"5.Director\n";
    cout<<"Enter the choice of your designation"<<endl;
    cin>>opt;
    cout<<"enter name\n";
    cin>>name;
    cout<<"enter userid\n";
    cin>>id;
    if(cin.fail())
    {
        cin.clear();
        cin.ignore(std::numeric_limits<std::streamsize>::max(),'\n');
        //if the choice entered is not an integer the situation is
        handled using cin.fail() ad cin.clear()
    }
    /* registers the user based on
    designation and also makes sure
    that the id used for registration
    is non existing
    */
    if(opt==1)
    {
        if(!_database.findug(id))
        {
            cout<<"Id already exists\n";return;
        }
        desig= "Undergrads";
        double income;
        cout<<"enter family income\n";
        cin>>income;
        Undergrads obj(name,id,desig);
        obj.setincome(income);
        _database.addug(obj);
    }
```

```
else if(opt==2)
{
    if(!_database.findres(id))
    {
        cout<<"Id already exists\n";
        return;
    }
    desig="Research_scholar";
    double income, facid;
    cout<<"enter family income\n";
    cin>>income;
    cout<<"enter faculty id under which you are working\n";
    cin>>facid;
    if(_database.findfac(facid))
    {/* makes sure that the faculty under which
        he works exists in the institute
    */
        cout<<"Concerned faculty not found..retry\n";
        return;
    }
    Research_scholar obj(name,id,desig,facid);
    obj.setincome(income);
    _database.addres(obj);
}
else if(opt==3)
{
    if(!_database.findfac(id))
    {
        cout<<"Id already exists\n";
        return;
    }
    desig="Faculty";
    string department;
    cout<<"Enter your department\n";
    cin>>department;
```

```
        if(_database.findhoddep(department))
        {/* makes sure that the hod under which
            he works exists in the institute
        */
            cout<<"Concerned Hod not found...retry\n";
            return;
        }
        Faculty obj(name,id,desig,department);
        _database.addfac(obj);
    }
    else if(opt==4)
    {
        if(!_database.findhod(id))
        {
            cout<<"Id already exists\n";
            return;
        }
        desig="Hod";
        string department;
        cout<<"Enter your department\n";
        cin>>department;
        Hod obj(name,id,desig,department);
        _database.addhod(obj);
    }
    else if(opt==5)
    {
        desig="Director";
        if(id!=_director.get_id())
        {
            cout<<"Wrong id..director exists\n";
            return;
        }
        Director obj(name,id,desig);
        _database.adddir(obj);
    }
    else
    {
        cout<<"Invalid choice..retry\n";
    }
}
```

3.15.4 Test cases

Test ID	User	Inputted ID	Expected Result	Actual Result	Result (Pass/ Fail)	Comments
-	**Administrator**	**1234 (default)**	-	-	-	-
-	**Director**	**123 (default)**	-	-	-	-
3.15.4.1	Undergrads	1	Registration Successful		**PASS**	-
3.15.4.2	Research_ scholar	2 (faculty ID : 5)	Registration Failed		**PASS**	Faculty not found
3.15.4.3	Faculty	3 (Department : CSE)	Registration Failed		**PASS**	HOD not found
3.15.4.4	HOD	4 (Department : CSE)	Registration Successful		**PASS**	-
3.15.4.5	Faculty	3 (Department : EE)	Registration Failed		**PASS**	HOD not found
3.15.4.6	Faculty	3 (Department : CSE)	Registration Successful		**PASS**	-
3.15.4.7	HOD	4 (Department : EE)	Registration Failed		**PASS**	User ID in use
3.15.4.8	HOD	5 (Department : EE)	Registration Successful		**PASS**	-
3.15.4.9	Faculty	6 (Department : EE)	Registration Successful		**PASS**	-
3.15.4.10	Research_ scholar	2 (faculty ID : 3)	Registration Successful		**PASS**	-
3.15.4.11	Research_ scholar	7 (faculty ID : 6)	Registration Successful		**PASS**	-

3.15.5 Test Result

The function get_status produced the output as expected for all the test cases.

The 2 criteria:

- Uniqueness of ID and
- Hierarchical Registration

are satisfied

3.16 LOGIN

Function Name	Login
Class	Utility_class
Input Parameters to the function	None
User inputs	Choice of designation (from a menu)
	ID
Processing	○ Checks if ID is already registered ○ If ID is registered, the control is handed over to the interface for the corresponding user.
Output Parameters	None

3.16.1 Test items

The unit to be tested here is the function Login (which facilitates the process of logging in).

3.16.2 Features to be tested

The system should allow a user to login, only if he has registered beforehand.

3.16.3 Item pass/ fail criteria

If the system allows a user to login, only if he has registered and generates an error message when the user is not registered, the test is deemed to have been passed.

3.16.4 Test Cases

Present Situation of the Database	
User ID	**Designation**
123	Director
1234	Administrator
1	HOD (CSE)
2	HOD (EE)
3	Faculty (CSE)
4	Faculty (EE)
5	Research_scholar (Faculty: 3)
6	Research_scholar (Faculty: 4)
7	Undergrads

Test ID	Inputted Designation	Inputted ID	Expected Result	Actual Result	Result (Pass/ Fail)	Comments
3.16.4.1	Administrator	1234	Login Successful	Login Successful	**PASS**	-
3.16.4.2	Administrator	1	Login Failed	Login Failed	**PASS**	Incorrect ID
3.16.4.3	Director	123	Login Successful	Login Successful	**PASS**	-
3.16.4.4	Director	1	Login Failed	Login Failed	**PASS**	Incorrect ID
3.16.4.5	HOD	1	Login Successful	Login Successful	**PASS**	-
3.16.4.6	HOD	2	Login Successful	Login Successful	**PASS**	-
3.16.4.7	HOD	3	Login Failed	Login Failed	**PASS**	Incorrect ID
3.16.4.8	Faculty	3	Login Successful	Login Successful	**PASS**	-
3.16.4.9	Faculty	4	Login Successful	Login Successful	**PASS**	-
3.16.4.10	Faculty	5	Login Failed	Login Failed	**PASS**	Incorrect ID
3.16.4.11	Research_ scholar	5	Login Successful	Login Successful	**PASS**	-
3.16.4.12	Research_ scholar	6	Login Successful	Login Successful	**PASS**	-
3.16.4.13	Research_ scholar	7	Login Failed	Login Failed	**PASS**	Incorrect ID
3.16.4.14	Undergrads	7	Login Successful	Login Successful	**PASS**	-
3.16.4.15	Undergrads	8	Login Failed	Login Failed	**PASS**	Incorrect ID

3.16.5 Test Result

The function Login produced results as expected.

Code Snippet

```
void Utility_class::login()
{   //interface for logging in which verifies your designation and id if they are registered
    string desig;
    double id;
    int opt;
    cout<<"Login as\n"<<"1.Undergrads\n"<<"2.Research scholar\n"<<
    "3.Faculty\n"<<"4.Hod\n"<<"5.Director\n"<<"6.Administrator\n";
    cout<<"Enter the choice of your designation"<<endl;
    cin>>opt;
    cout<<"Enter your id\n";
    cin>>id;
    if(cin.fail())
    {
      cin.clear();
      cin.ignore(std::numeric_limits<std::streamsize>::max(),'\n');
    }
    if(opt==1)
    {
        try
        {
            Undergrads obj=_database.get_ug(id);
            this->visitug(obj);
        }
        catch(...)
        {
            cout<<"Id Not found\n";
        }
    }
    else if(opt==2)
    {
        try
        {
            Research_scholar obj=_database.get_res(id);
            this->visitres(obj);
        }
        catch(...)
        {
            cout<<"Id Not found\n";
        }
    }
    else if(opt==3)
    {
        try
        {
```

```
            Faculty obj=_database.get_fac(id);
            this->visitfac(obj);
        }
        catch(...)
        {
            cout<<"Id Not found\n";
        }
    }
    else if(opt==4)
    {
        try
        {
            Hod obj=_database.get_hod(id);
            this->visithod(obj);
        }
        catch(...)
        {
            cout<<"Id Not found\n";
        }
    }
    else if(opt==5)
    {
        try
        {
            Director obj=_database.get_dir(id);
            this->visitdir(obj);
        }
        catch(...)
        {
            cout<<"Id Not found\n";
        }
    }
    else if(opt==6)
    {
        if(id==_administrator.get_id())
            this->visitadm();
        else
        {
            cout<<"Id Not found\n";
        }
    }
    else   cout<<"Invalid choice..retry\n";
}
```

3.17

4 DETAILS OF SYSTEM TESTING

This is the Testing performed on the Application under test

- To make sure that Critical Business scenarios were tested.
- To verify the entire application works as per the requirements.
- To verify that functionalities in the application work as intended without any errors.

- **Entry Criteria:**

1. The unit testing is performed for each module
2. For each user we take, we consider the case that he has already registered and hence logged in.

- **Exit criteria:**

1. All test cases are executed.
2. All defects if existent are removed.
3. The functionalities expected from the software are executing without failure.

4.1 LEAVE PROCESSING:

UNDERGRAD TO ADMINISTRATOR (END-TO-END)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	UG: apply leave **Example test case:** Undergrads(mode): Id:2 Days entered:20	1. Days: number of leaves applied 2. Leave: leave object associated with Undergrad	1. Status: status of undergrad changes from "Not applied" to "Administrator" if Days<max_leaves_premitted 2. The leave object of the applicant is inserted in the list of leaves pending which needs to be approved by administrator	1. Same as expected result 2. Same as expected result	Pass Pass	The leave application applied by undergrad is processed to Administrator if it satisfies the conditions as mentioned in unit testing.
2	Administrator: approve leave **Example test case:** Administrator(mode) For approval: Selected id:2 Selected designation: Undergrads	Id of leave applicant Designation of leave applicant	1. Status: status of undergrad changes from "Administrator" to "Approved" 2. The leave application object is removed from leave pending list of administrator 3. Max leaves: of the applicant now reduces from initial number by the amount of leaves which he has applied and are now approved	1. Same as expected 2. Same as expected 3. Same as expected	Pass Pass 3.Pass	Once the administrator approves the concerned leave application request of applicant, leave is approved and maximum leaves remaining are deducted

4.2 LEAVE PROCESSING:

RESEARCH SCHOLAR TO ADMINISTRATOR (END-TO-END AND PAIRWISE)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	Research_scholar: apply leave **Example test case:** Reserach_scholar (mode): Id:12 Days entered:10	1. Days: number of leaves applied 2. Leave: leave object associated with research_scholar	1. Status: status of research scholar changes from "Not applied" to "Faculty" if Days < max_leaves_premitted 2. The leave object of the applicant is inserted in the list of leaves pending which needs to be approved by faculty advisor of the research scholar	1. Same as expected result 2. Same as expected result	Pass Pass	The leave application of a research scholar is forwarded to his faculty advisor if it satisfies the conditions as mentioned in unit testing.
2	Faculty: approve leave (Faculty who is the advisor of concerned research scholar) **Example test case:** Faculty(mode) For approval: Selected id:12 Selected designation: Research_scholar	1. Id of leave applicant 2. Designation of applicant	1. Status: status of research scholar changes from "Faculty" to "Administrator" 2. The leave application object is removed from leave pending list of faculty advisor and inserted in the list of leaves pending which needs to be approved by administrator	1. Same as expected 2. Same as expected	Pass Pass	Once the faculty advisor approves the application of applicant, leave is processed and forwarded to list of pending leaves of administrator
3	Administrator: approve leave **Example test case:** Administrator(mode) For approval: Selected id:12 Selected designation: Research_scholar	1. Id of leave applicant 2. Designation of leave applicant	1. Status: status of scholar changes from "Administrator" to "Approved" 2. The leave application object is removed from leave pending list of administrator 3. Max leaves: of the applicant now reduces from initial number by the amount of leaves which he has applied and are now approved	1. Same as expected 2. Same as expected 3. Same as expected	Pass Pass 3.Pass	Once the administrator approves the concerned leave application request of applicant, leave is approved and maximum leaves remaining are deducted

4.3 LEAVE PROCESSING:

FACULTY TO ADMINISTRATOR (END-TO-END AND PAIRWISE)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	Faculty: apply leave **Example test case:** Faculty(mode): Id:1 Days entered:15	1. Days: number of leaves applied 2. Leave: leave object associated with faculty	1. Status: status of faculty changes from "Not applied" to " HOD " if Days<max_leaves_premitted 2. The leave object of the applicant is inserted in the list of leaves pending which needs to be approved by HOD of the faculty	1. Same as expected result 2.Same as expected result	Pass Pass	The leave application applied by faculty is processed to his HOD if it satisfies the conditions as mentioned in unit testing.
2	HOD: approve leave (HOD of department in which faculty works) **Example test case:** HOD (mode) For approval: Selected id:1 Selected designation: Faculty	1. Id of leave applicant 2. Designation of applicant	1. Status: status of faculty changes from " HOD " to "Administrator" 2. The leave application object is removed from the list of pending leaves of the HOD and inserted in the list of pending leaves which needs to be approved by administrator	1. Same as expected 2. Same as expected	Pass Pass	Once the HOD approves the concerned leave application request of applicant, leave is processed forward to list of pending leaves of administrator
3	Administrator: approve leave **Example test case:** Administrator(mode) For approval: Selected id:1 Selected designation: Faculty	1. Id of leave applicant 2. Designation of leave applicant	1. Status: status of faculty changes from "Administrator" to "Approved" 2. The leave application object is removed from leave pending list of administrator 3. Max leaves: of the applicant now reduces from initial number by the amount of leaves which he has applied and are now approved	1. Same as expected 2. Same as expected 3.Same as expected	Pass Pass 3.Pass	Once the administrator approves the concerned leave application request of applicant, leave is approved and maximum leaves remaining are deducted

4.4 LEAVE PROCESSING:

HOD TO ADMINISTRATOR (END-TO-END AND PAIRWISE)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	HOD: apply leave **Example test case:** HOD (mode): Id:5 Days entered:5	1. Days :number of leaves applied 2. Leave: leave object associated with HOD	1. Status: status of undergrad changes from "Not applied" to "Director" if Days < max_leaves_premitted 2. The leave object of the applicant is inserted in the list of leaves pending which needs to be approved by the Director	1. Same as expected result 2.Same as expected result	Pass Pass	The leave application applied by HOD is processed to director if it satisfies the conditions as mentioned in unit testing.
2	Director: approve leave **Example test case:** Director (mode) For approval: Selected id:5 Selected designation: HOD	1. Id of leave applicant 2. Designation of applicant	1. Status: status of HOD changes from "Director" to "Administrator" 2. The leave application object is removed from leave pending list of Director and inserted in the list of leaves pending which needs to be approved by administrator	1. Same as expected 2. Same as expected	Pass Pass	Once the director approves the concerned leave application request of applicant, leave is processed forward to list of pending leaves of administrator
3	Administrator: approve leave **Example test case:** Administrator(mode) For approval: Selected id:12 Selected designation: HOD	1. Id of leave applicant 2. Designation of leave applicant	1. Status: status of scholar changes from "Administrator" to "Approved" 2. The leave application object is removed from leave pending list of administrator 3. Max leaves: of the applicant now reduces from initial number by the amount of leaves which he has applied and are now approved	1. Same as expected 2. Same as expected 3.Same as expected	Pass Pass 3.Pass	Once the administrator approves the concerned leave application request of applicant, leave is approved and maximum leaves remaining are deducted

4.5 LEAVE PROCESSING:

DIRECTOR TO ADMINISTRATOR(END-TO-END)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	Director: apply leave **Example test case:** Director(mode): Id:123(only possible id) Days entered:22	1. Days: number of leaves applied 2. Leave: leave object associated with Director	1. Status:status of director changes from "Not applied" to "Administrator" if Days < max_leaves_premitted 2. The leave object of the applicant is inserted in the list of leaves pending which needs to be approved by administrator	1. Same as expected result 2.Same as expected result	Pass Pass	The leave application applied by director is processed to Administrator if it satisfies the conditions as mentioned in unit testing.
2	Administrator: approve leave **Example test case:** Administrator(mode) For approval: Selected id:123 Selected designation: Director	1. Id of leave applicant 2. Designation of leave applicant	1. Status: status of director changes from "Administrator" to "Approved" 2. The leave application object is removed from leave pending list of administrator 3. Max leaves: of the applicant now reduces from initial number by the amount of leaves which he has applied and are now approved	1. Same as expected 2. Same as expected 3.Same as expected	Pass Pass 3.Pass	Once the administrator approves the concerned leave application request of applicant, leave is approved and maximum leaves remaining are deducted

4.6 FINANCE PROCESSING:

UNDERGRAD TO ADMINISTRATOR (END-TO-END)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	UG: apply finance **Example test case:** Undergrads(mode): Id:2 amount entered:2000	1. Amount: total financial assistance sought 2. Finance: finance object associated with Undergrad	1. Status: status of undergrad changes from "Not applied" to "Administrator" if amount applied + current scholarship < upper bound and family income is within given bracket 2. The finance object of the applicant is inserted in the list of finances pending which needs to be approved by administrator	1. Same as expected result 2. Same as expected result	Pass Pass	The finance application applied by undergrad is processed to Administrator if it satisfies the conditions as mentioned in unit testing.
2	Administrator: approve finance **Example test case:** Administrator(mode) For approval: Selected id:2 Selected designation: Undergrads	1. ID of finance applicant 2. Designation of finance applicant	1. Status: status of undergrad changes from "Administrator" to "Approved" 2. The finance application object is removed from the list of pending finances of administrator 3. Amount withdrawn: The amount received as financial assistance is updated by adding to it, the amount which has been approved now.	1. Same as expected 2. Same as expected 3. Same as expected	Pass Pass 3.Pass	Once the administrator approves the concerned finance application request of applicant, finance is approved and amount withdrawn is updated accordingly

4.7 FINANCE PROCESSING:

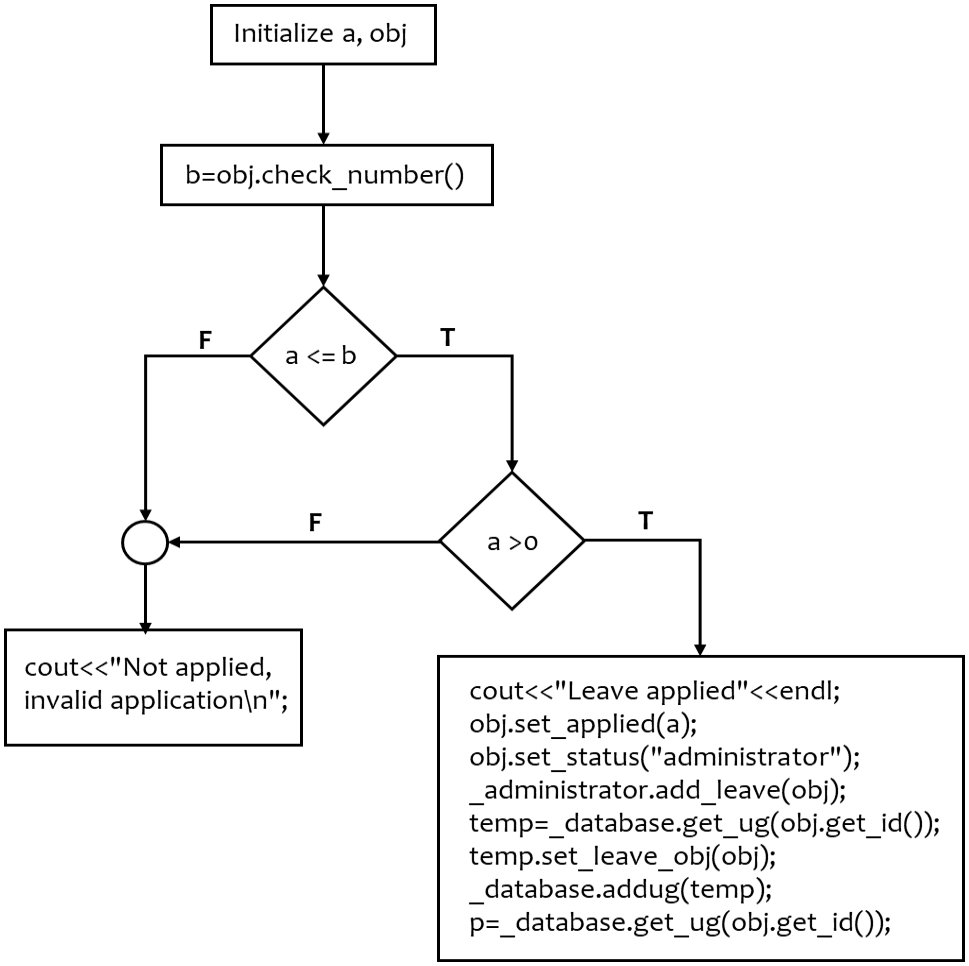
RESEARCH SCHOLAR TO ADMINISTRATOR (END-TO-END AND PAIRWISE)

Step	Test Steps	Inputs	Intended Result	Actual Result	Pass/Fail	Comments
1	Research_scholar: apply finance **Example test case:** Research_scholar(mode): Id:5 amount entered:5000	1. Amount :total financial assistance sought 2. Finance: finance object associated with Undergrad	1. Status: status of scholar changes from "Not applied" to "faculty if amount applied + current scholarship<upper bound and family income is within given bracket 2. The finance object of the applicant is inserted in the list of pending finances which need to be approved by Faculty	1. Same as expected result 2.Same as expected result	Pass Pass	The finance application applied by research scholar is processed to his faculty advisor if it satisfies the conditions as mentioned in unit testing..
2	Faculty: approve finance (concerned faculty advisor of the research scholar) **Example test case:** Faculty (mode) For approval: Selected id:5 Selected designation: Research_scholar	1. ID of finance applicant 2. Designation of finance applicant	1. Status: status of scholar changes from "faculty" to "Administrator". 2. The finance application object is removed from list of pending finances of faculty and inserted into the list of pending finances of administrator	1. Same as expected 2. Same as expected	Pass Pass	Once the faculty approves the application request, finance is processed and forwarded to the list of pending finances of administrator
3	Administrator: approve finance **Example test case:** Administrator(mode) For approval: Selected id:12 Selected designation: Research_scholar	1. ID of finance applicant 2. Designation of finance applicant	1. Status: status of scholar changes from "Administrator" to "Approved" 2. The finance application object is removed from the list of pending finances of administrator 3. Amount withdrawn: The amount received as financial assistance is updated by adding to it, the amount which has been approved now.	1. Same as expected 2. Same as expected 3.Same as expected	Pass Pass 3.Pass	Once the administrator approves the application request, finance is approved and amount withdrawn is updated accordingly

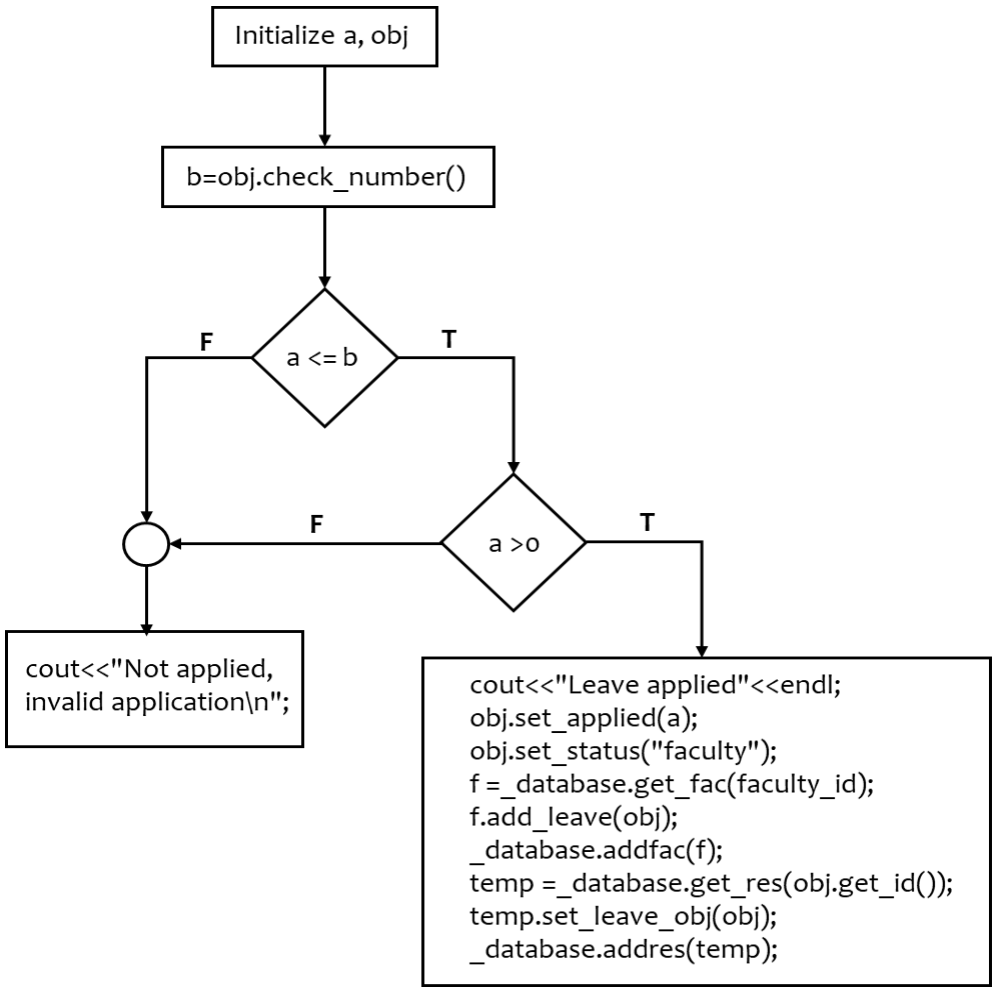
5 APPENDIX

Control Flow Graphs

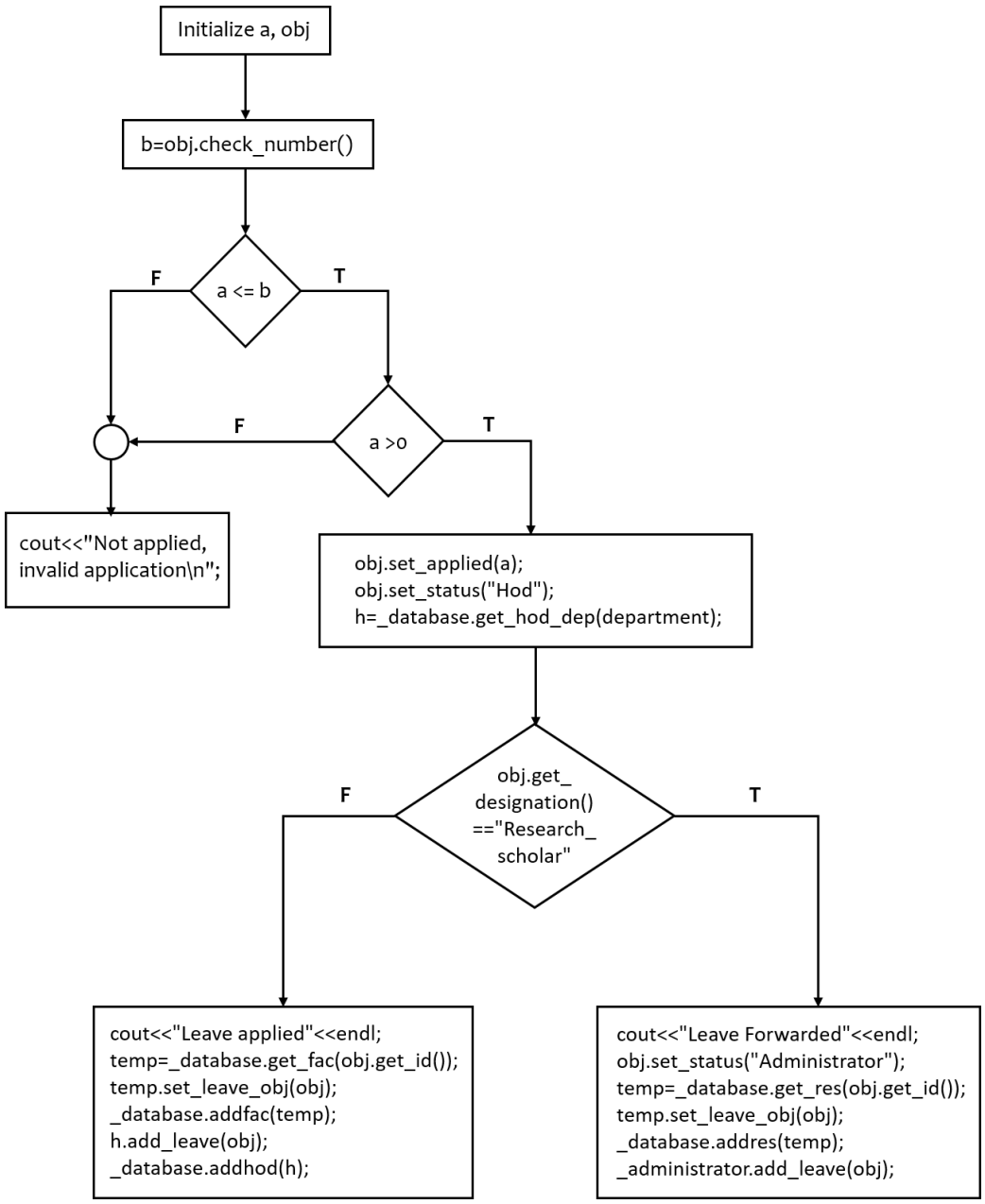
1. apply_leave for Undergrads



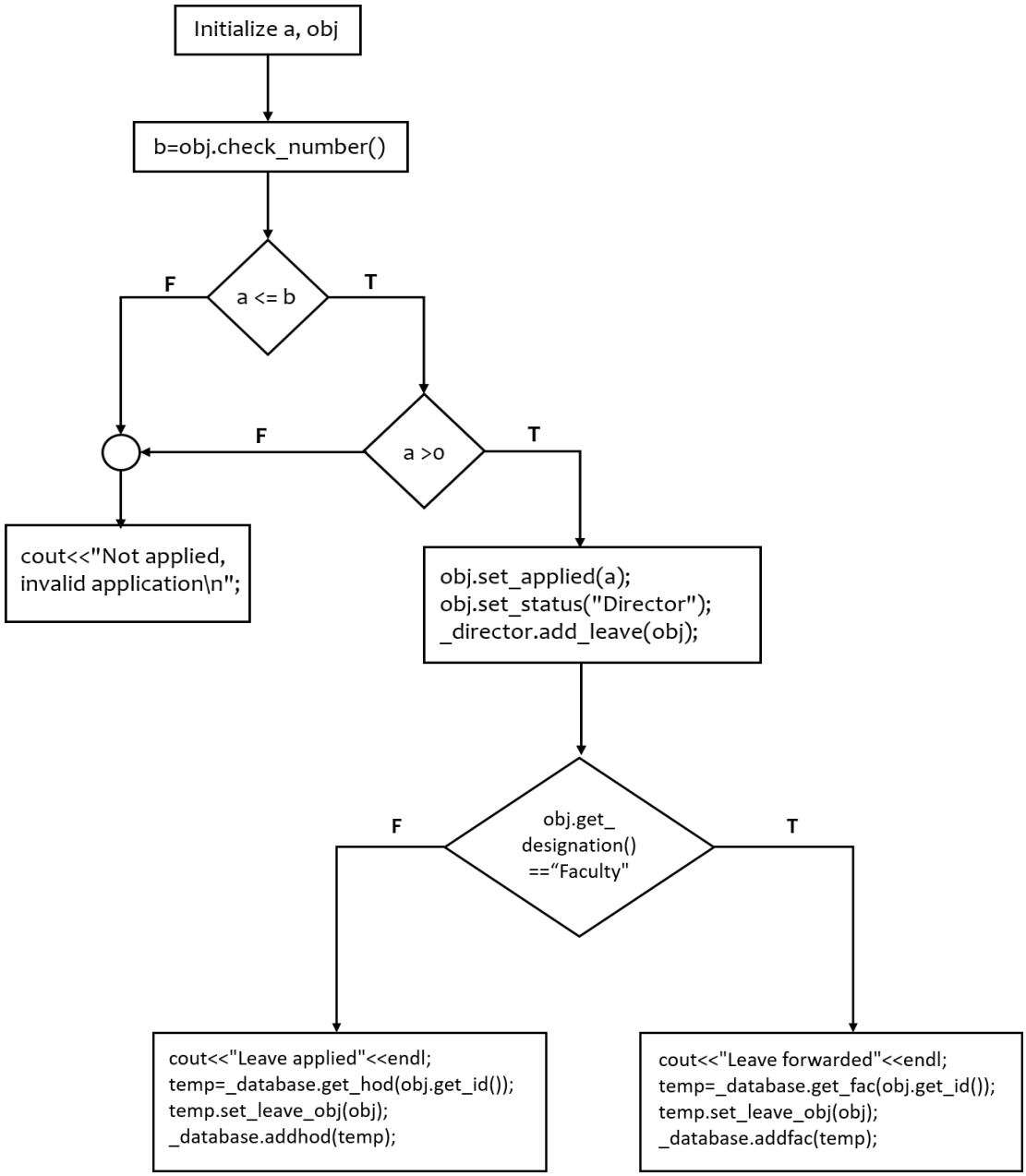
2. apply_leave for Research_scholar



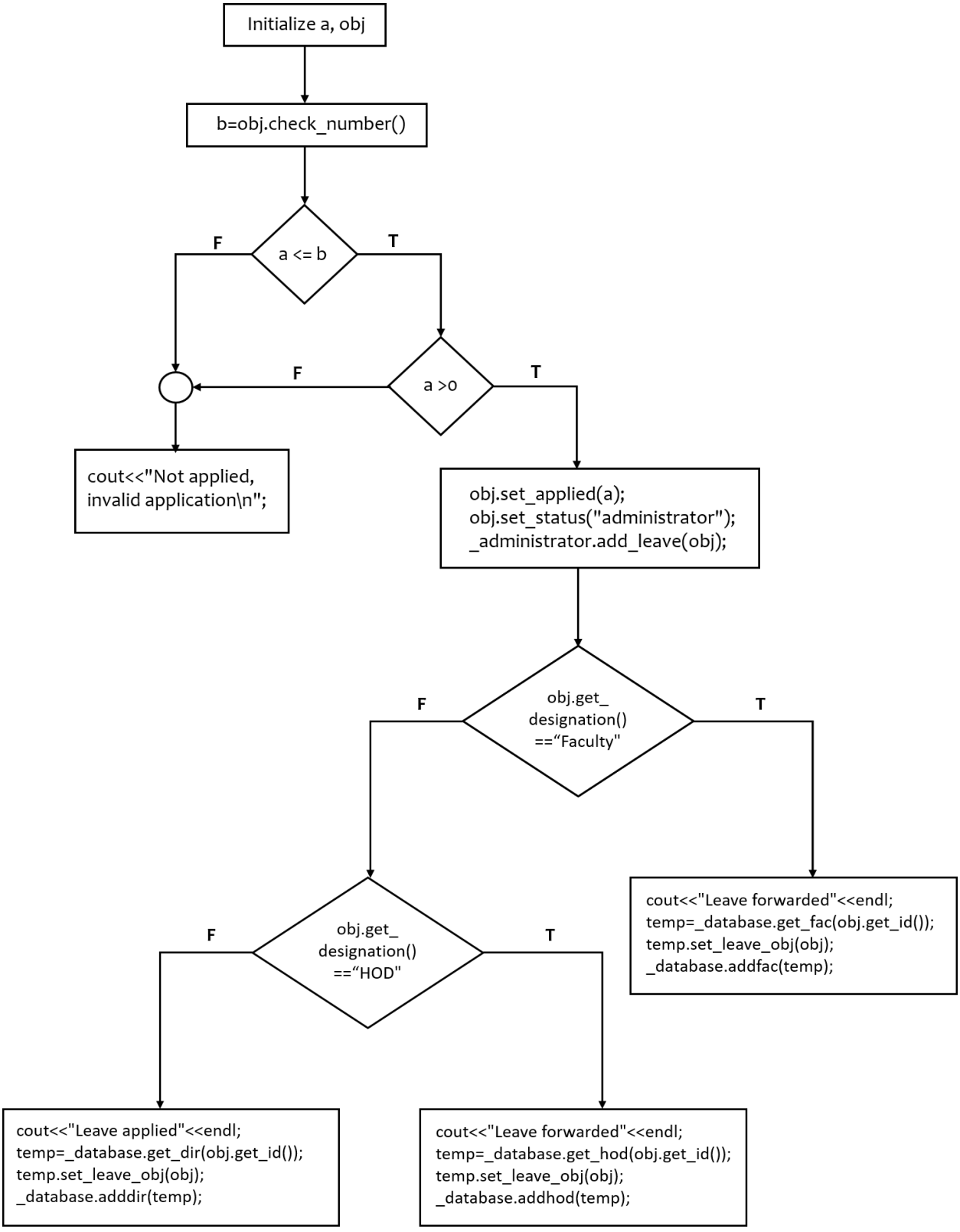
3. apply_leave for Faculty



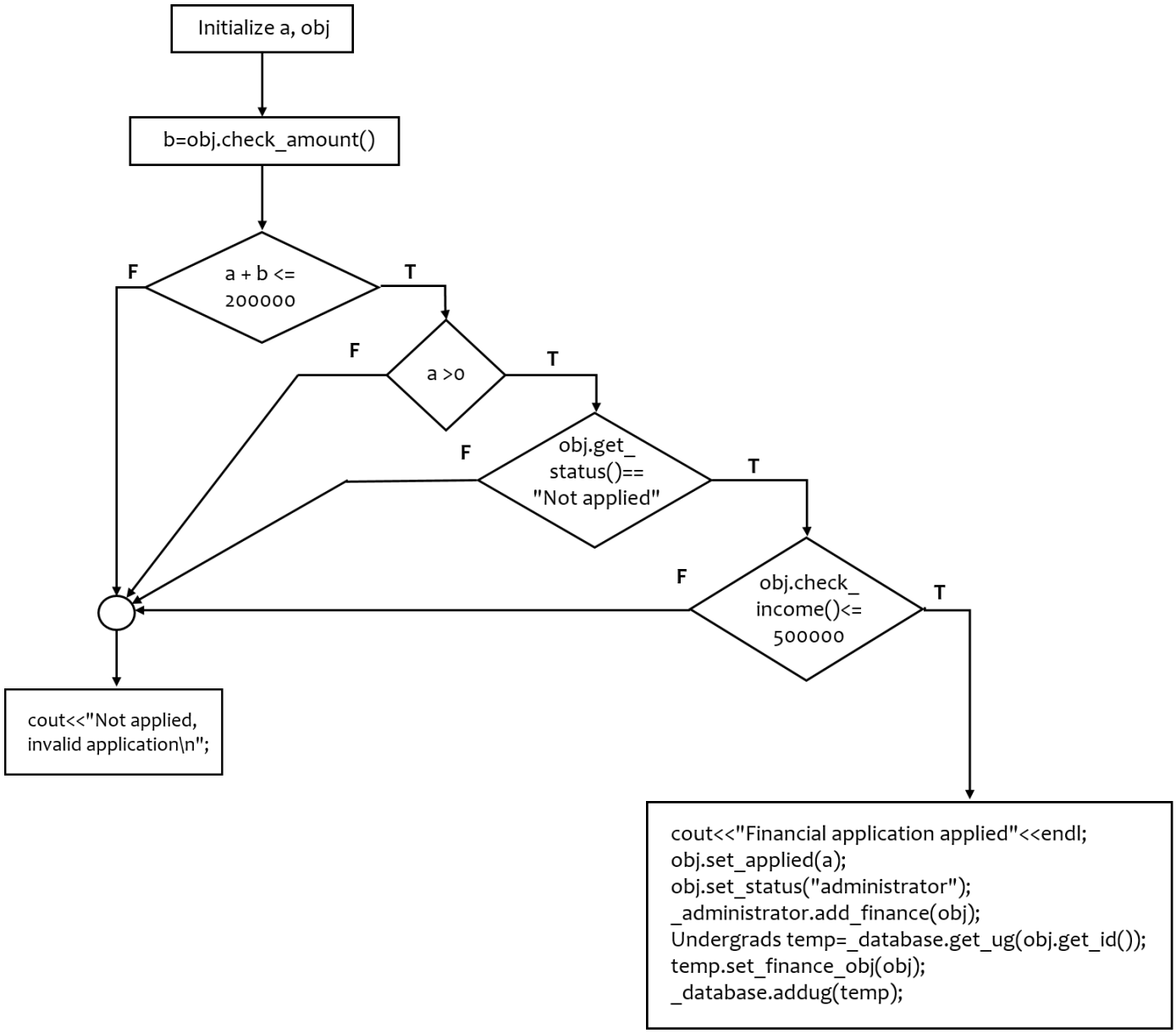
4. apply_leave for HOD



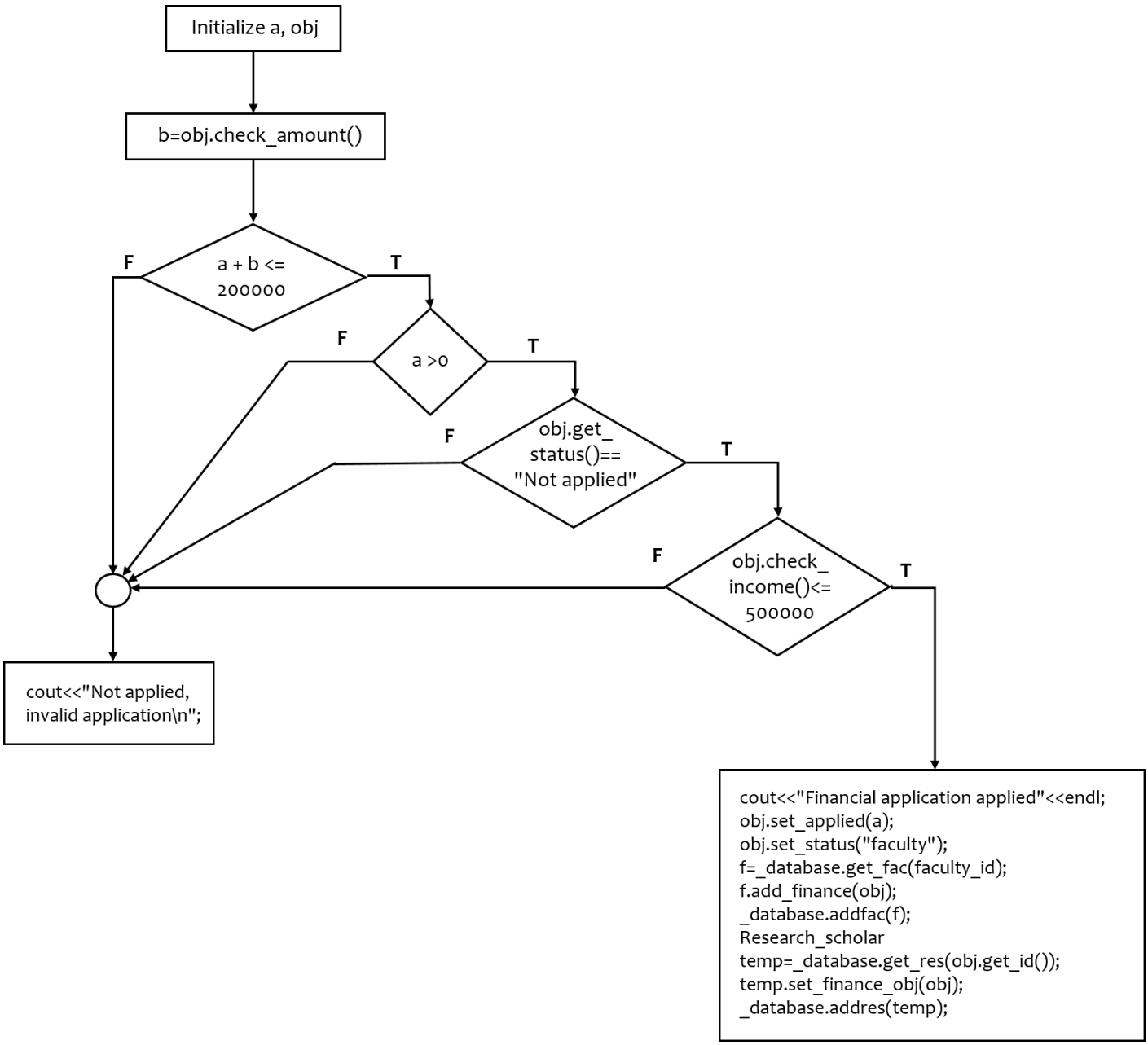
5. apply_leave for HOD



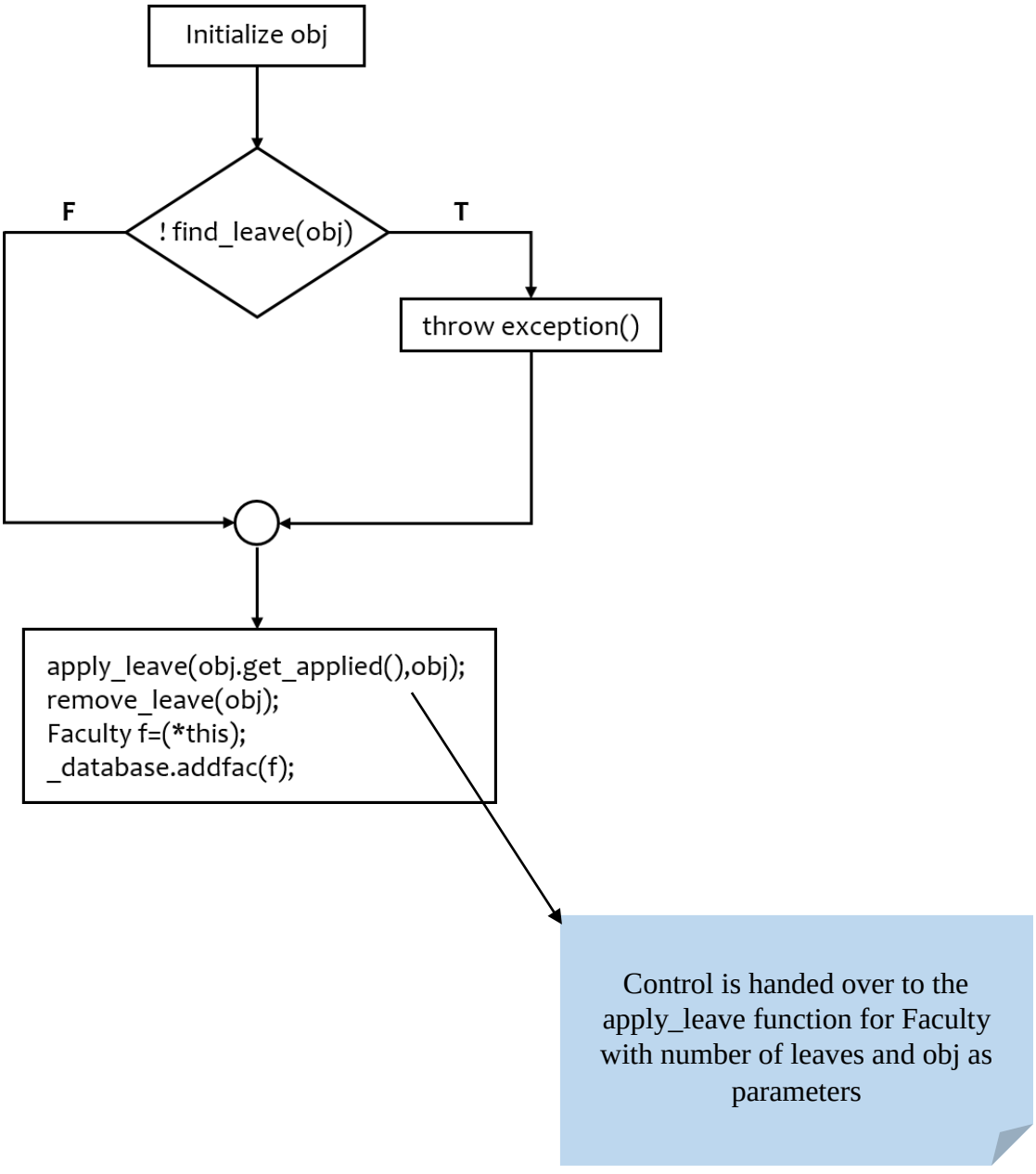
6. apply_finance for Undergrads



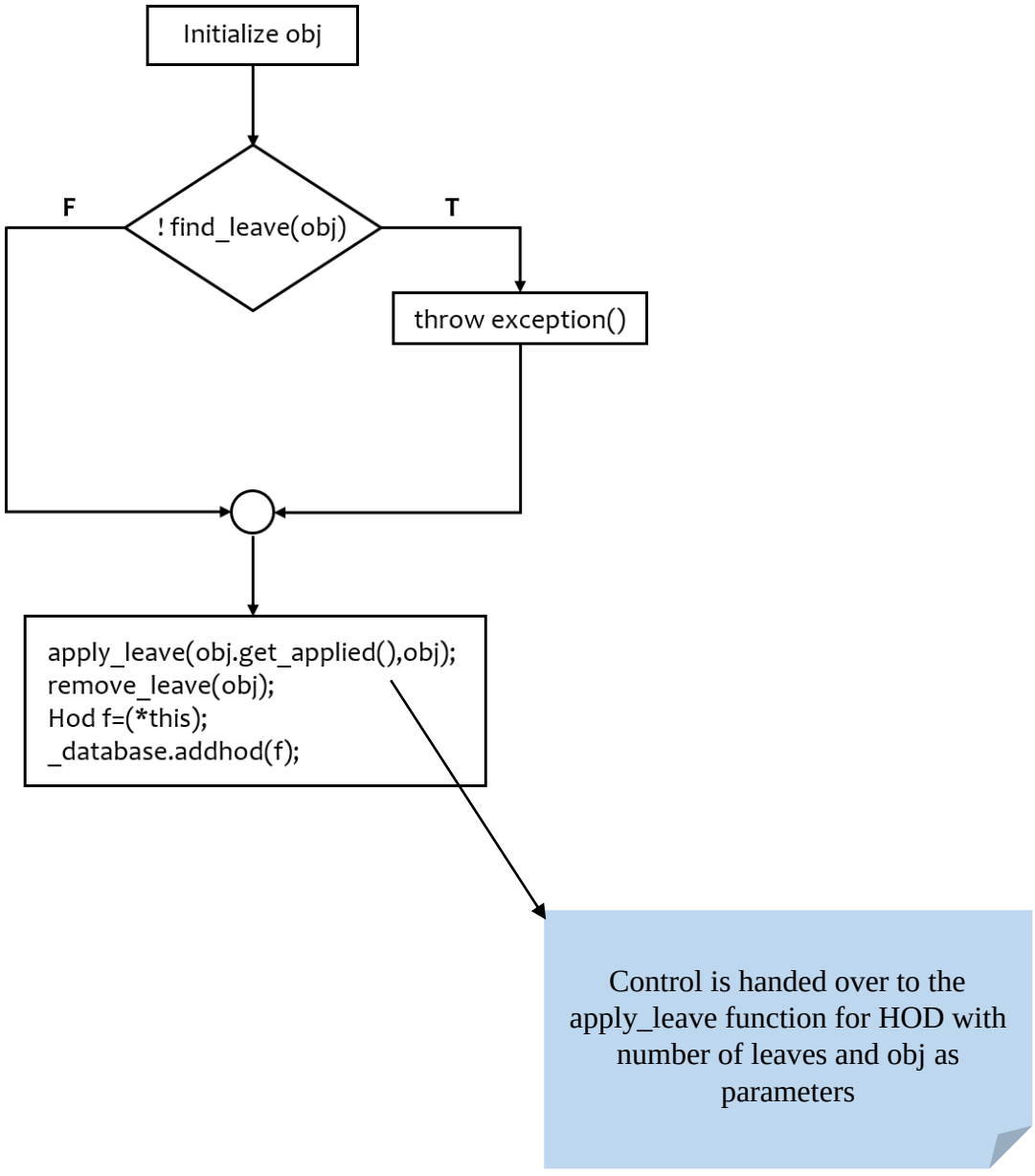
7. apply_finance for Research_scholar



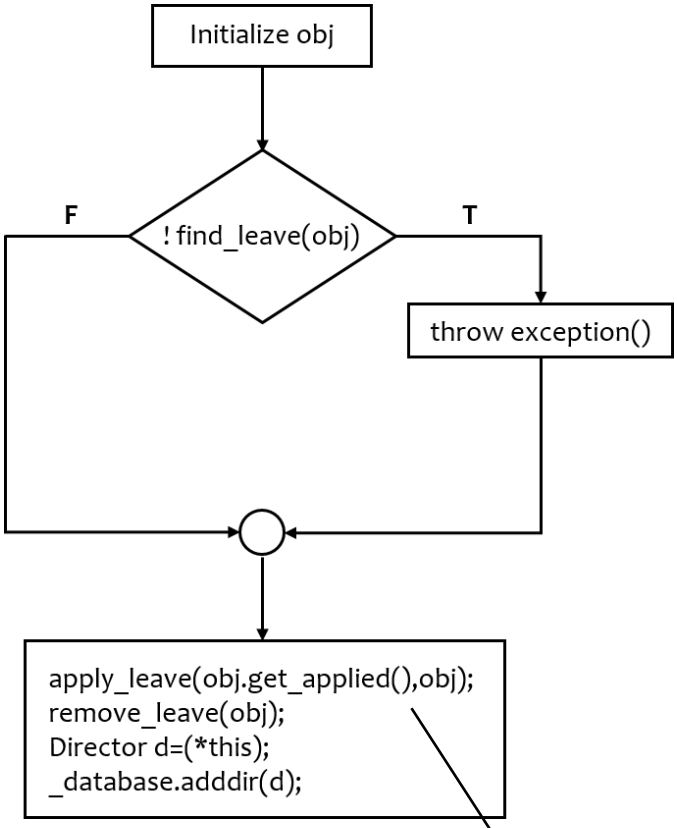
8. approve_leave for Faculty



9. approve_leave for HOD

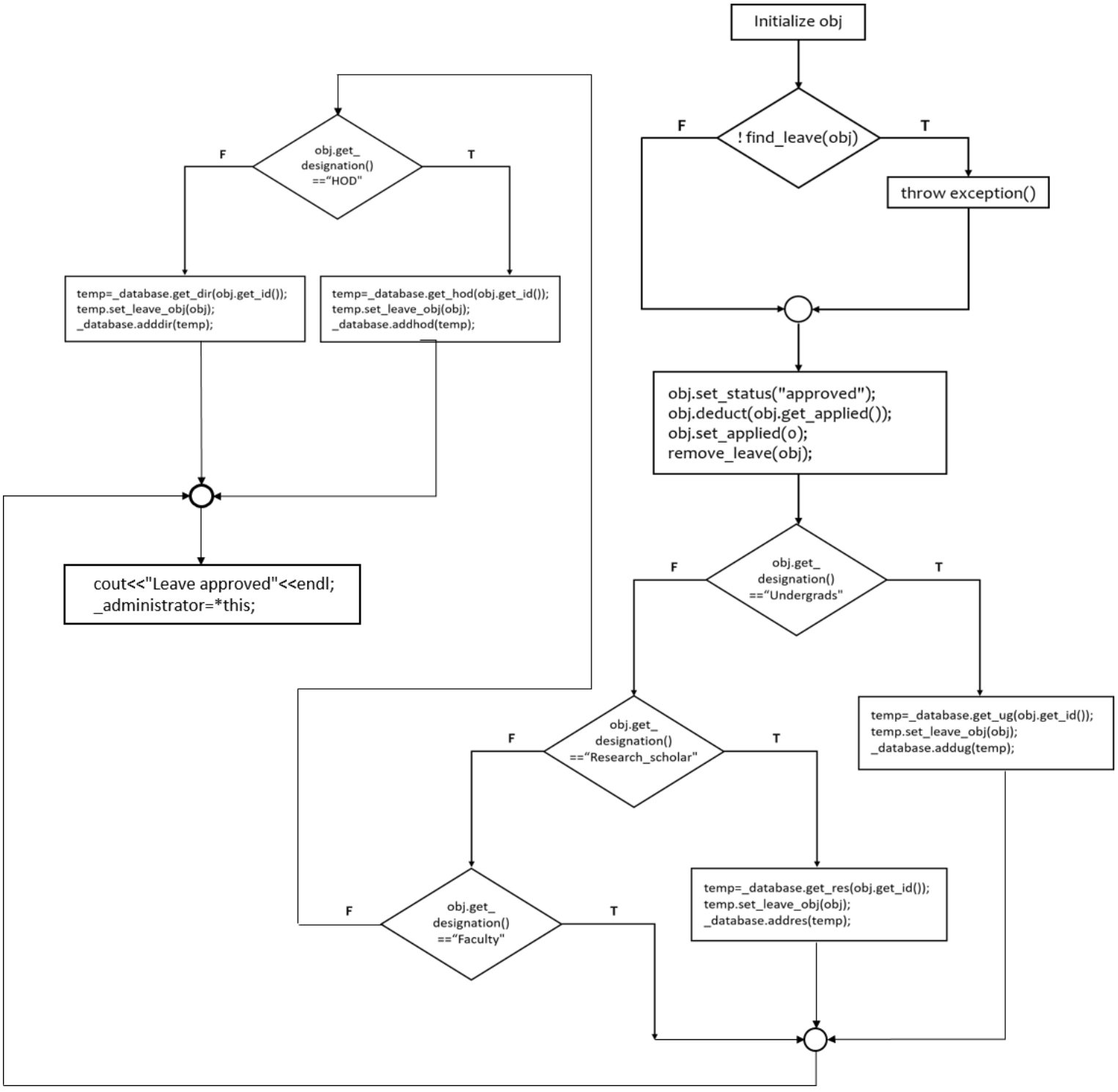


10. approve_leave for Director

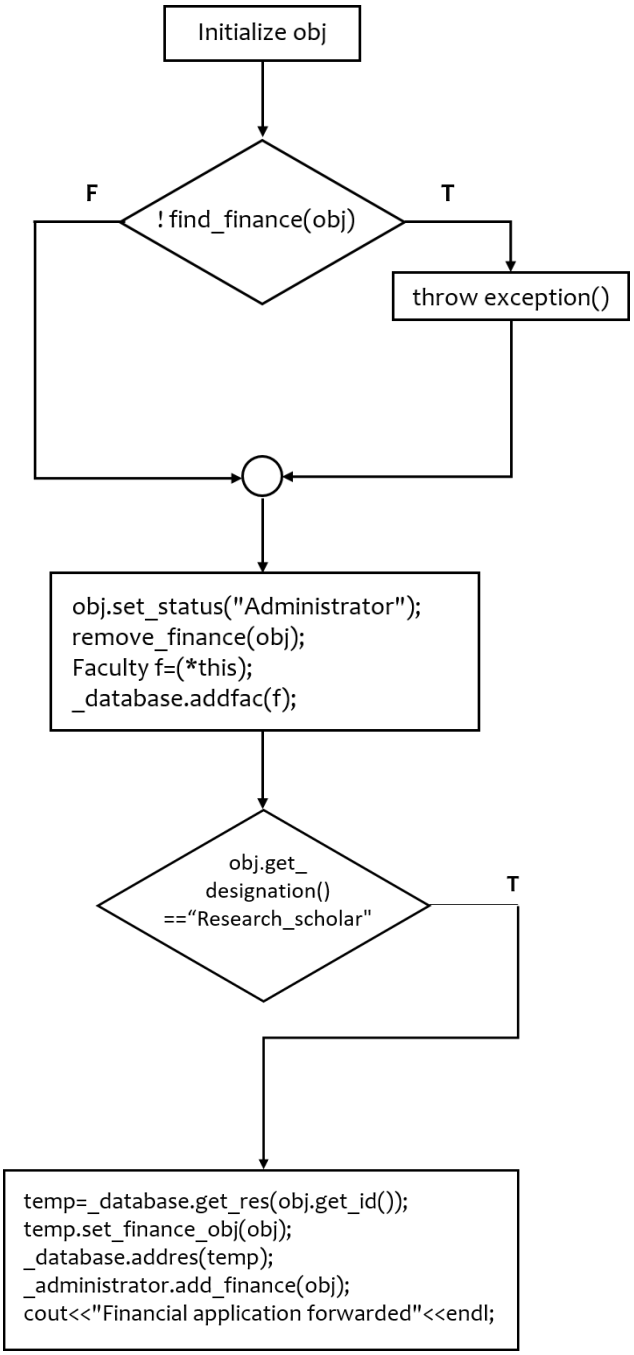


Control is handed over to the apply_leave function for the Director with number of leaves and obj as parameters

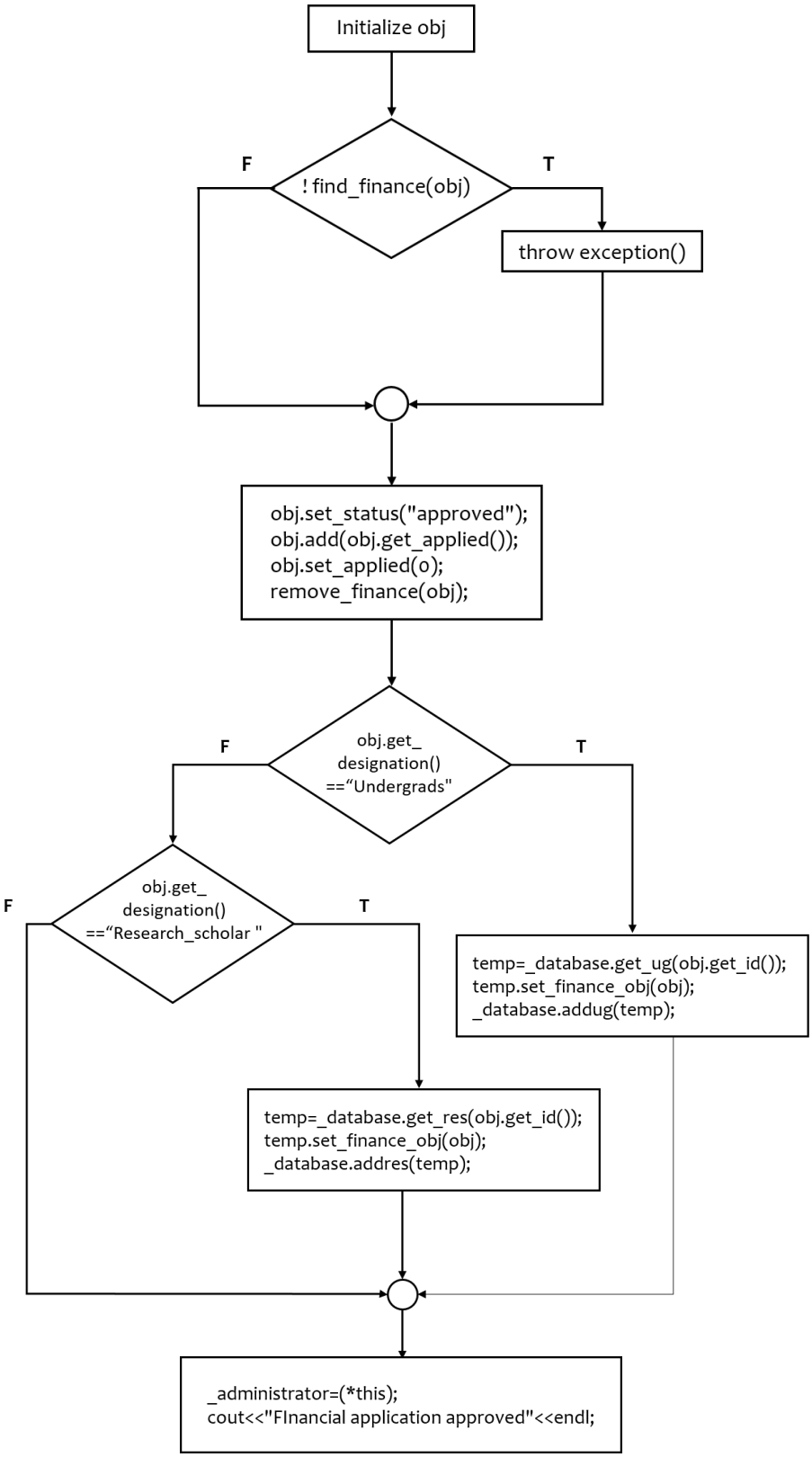
11. approve_leave for Administrator



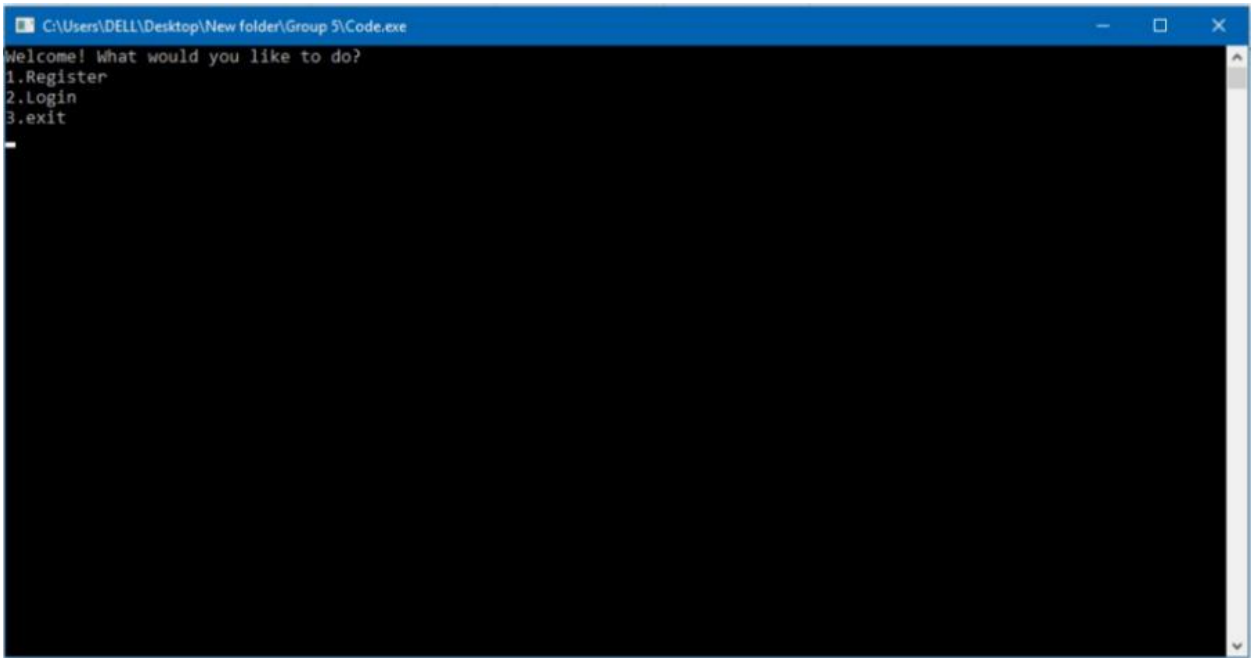
12. approve_finance for Faculty

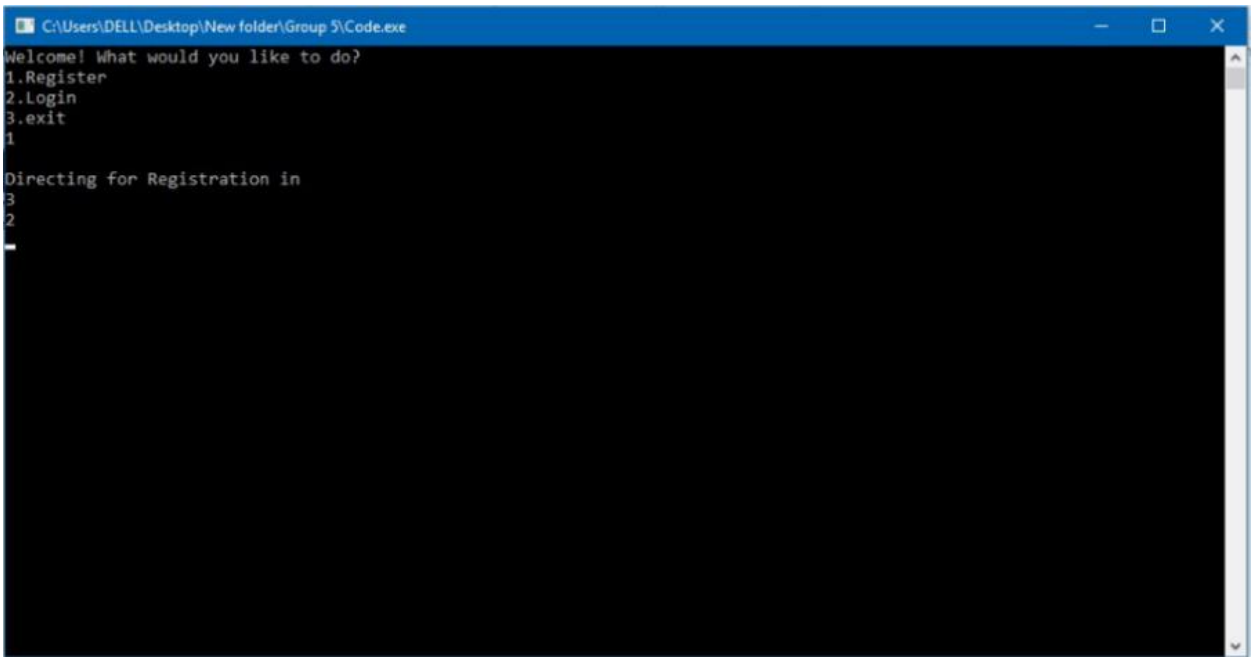


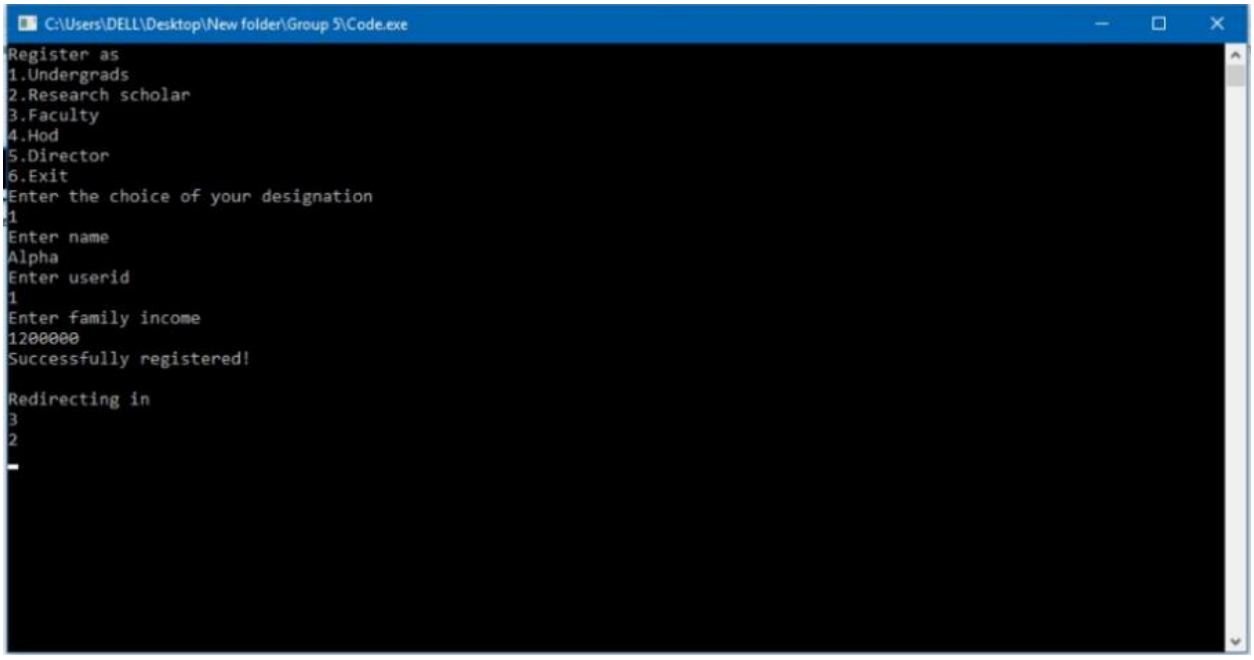
13. approve_finance for Administrator

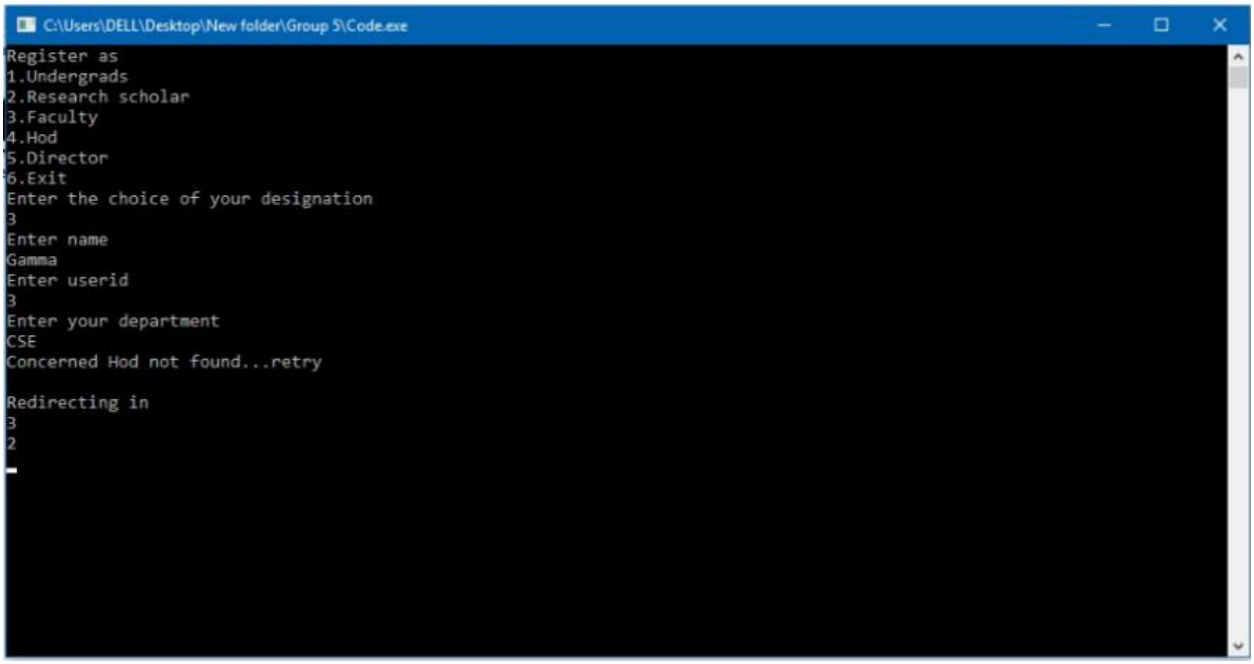


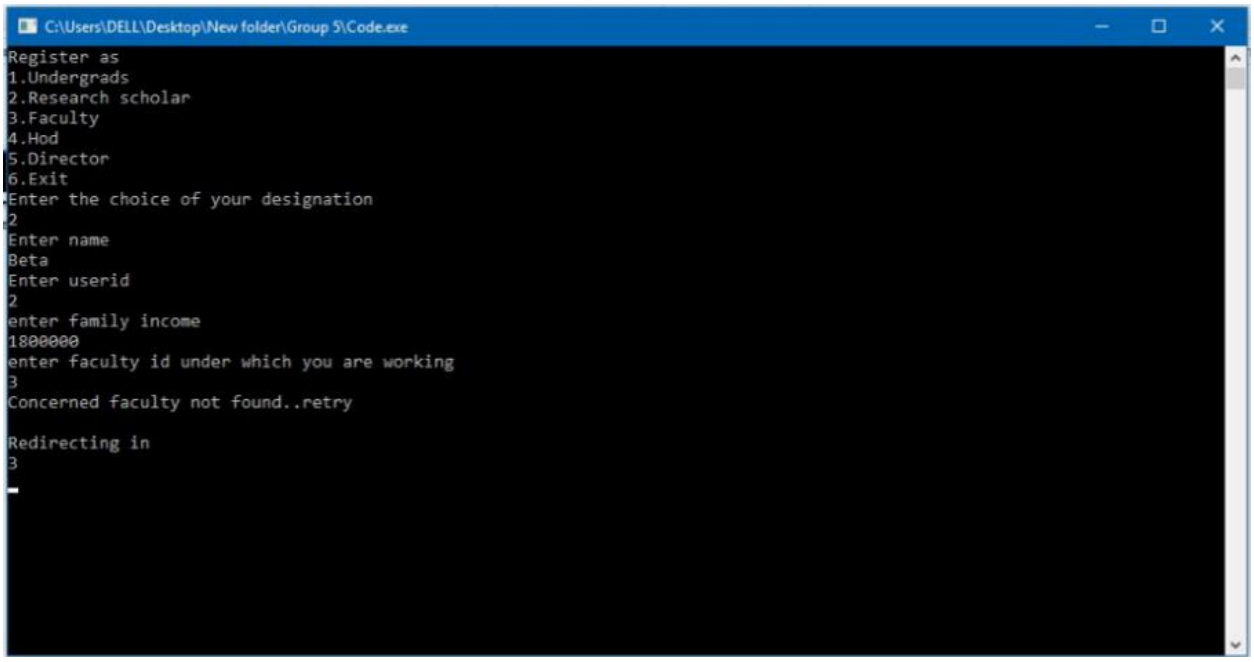
Glimpses into the execution …

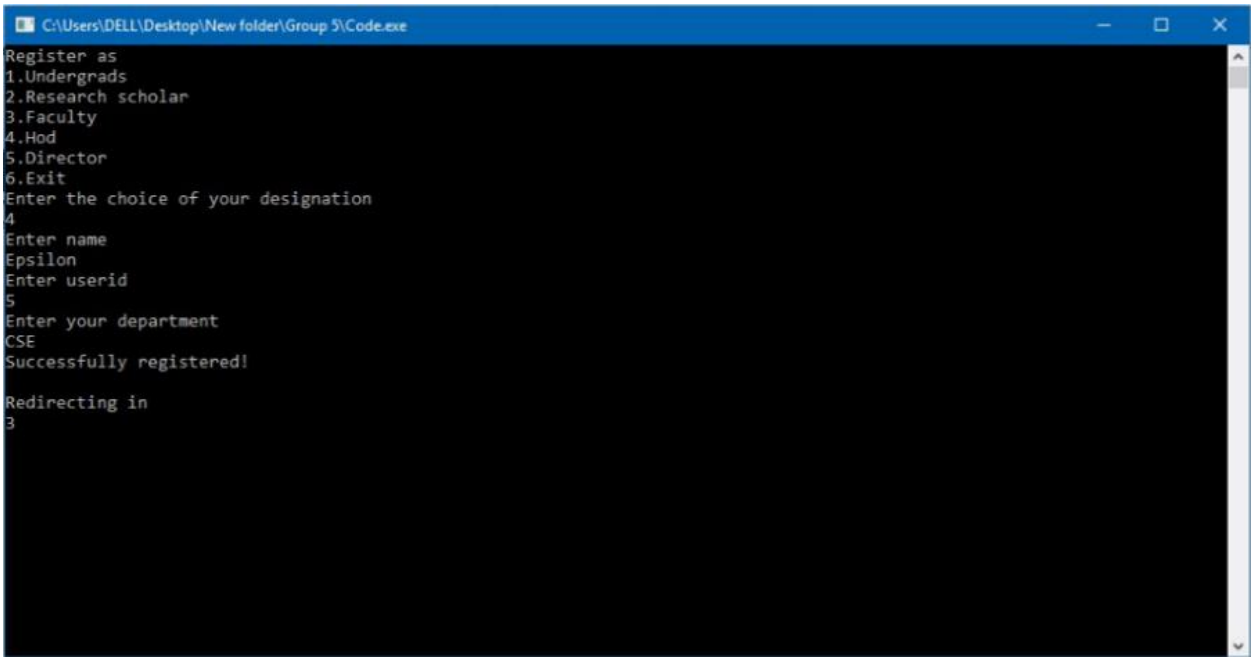


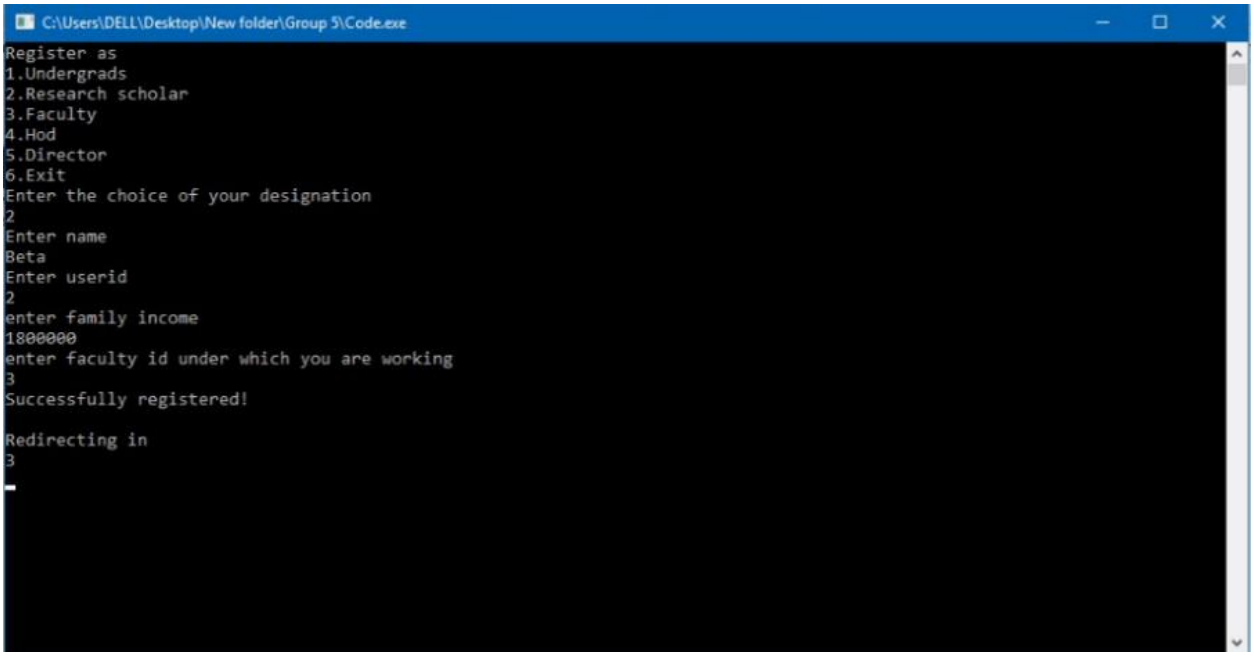


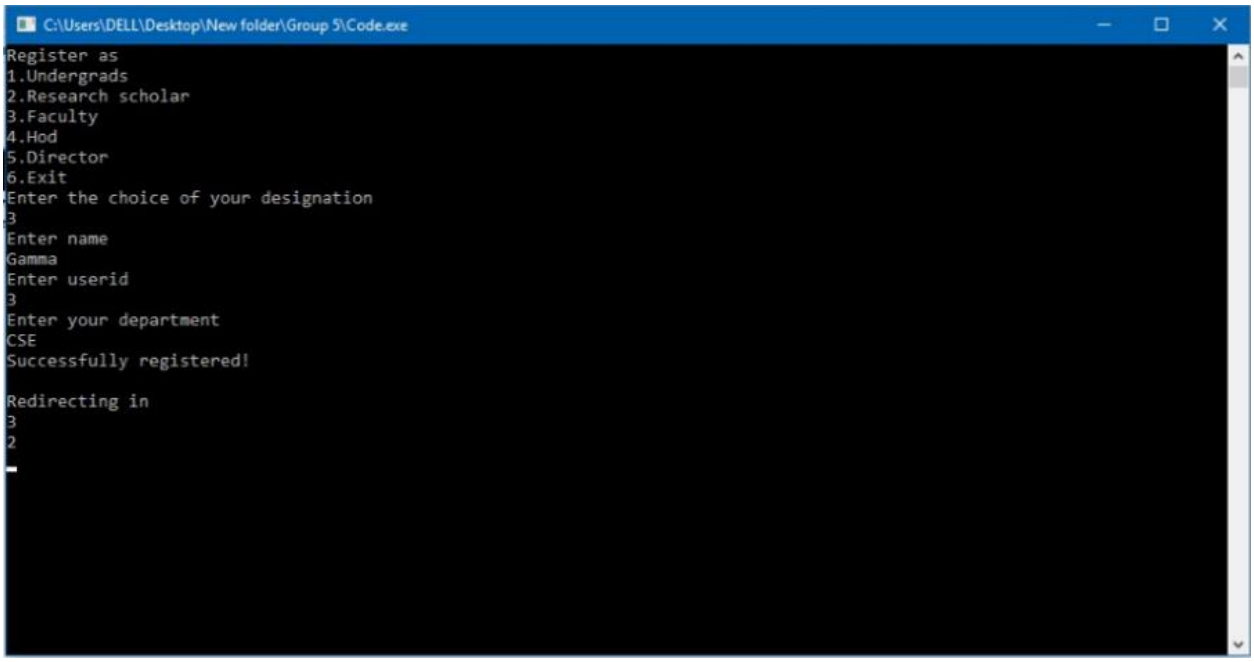

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Register as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Exit
Enter the choice of your designation
1
Enter name
Alpha
Enter userid
1
Enter family income
1200000
Successfully registered!
Redirecting in
3
2

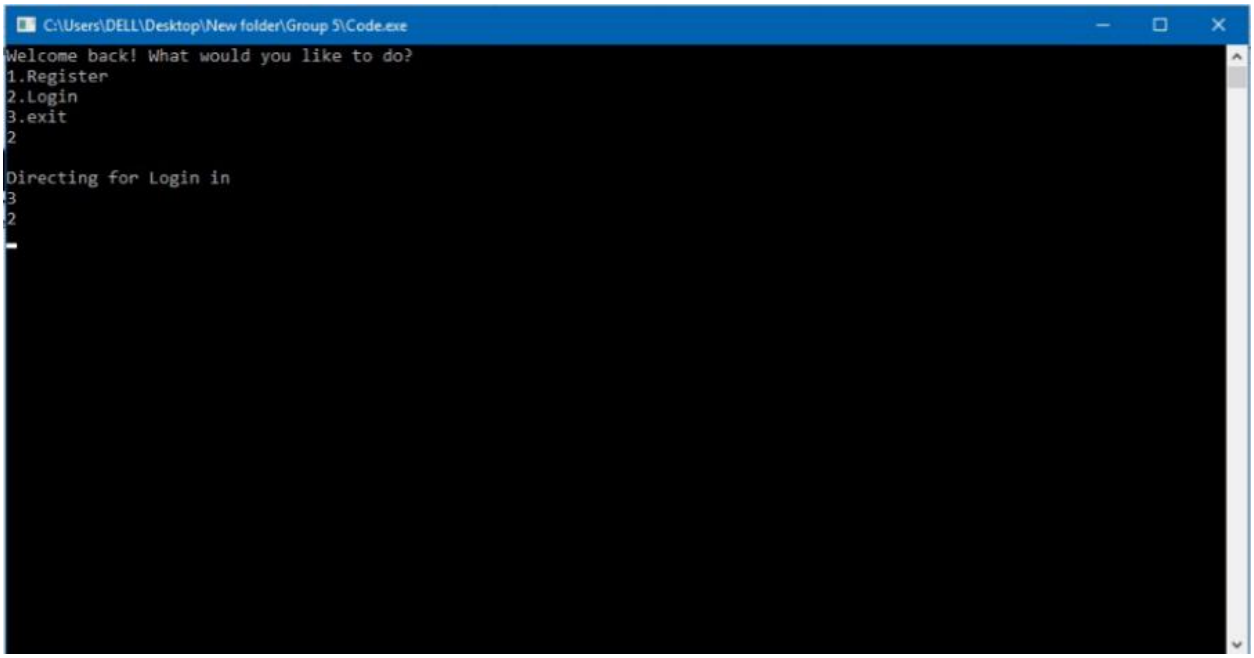

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Register as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Exit
Enter the choice of your designation
3
Enter name
Gamma
Enter userid
3
Enter your department
CSE
Concerned Hod not found...retry
Redirecting in
3
2

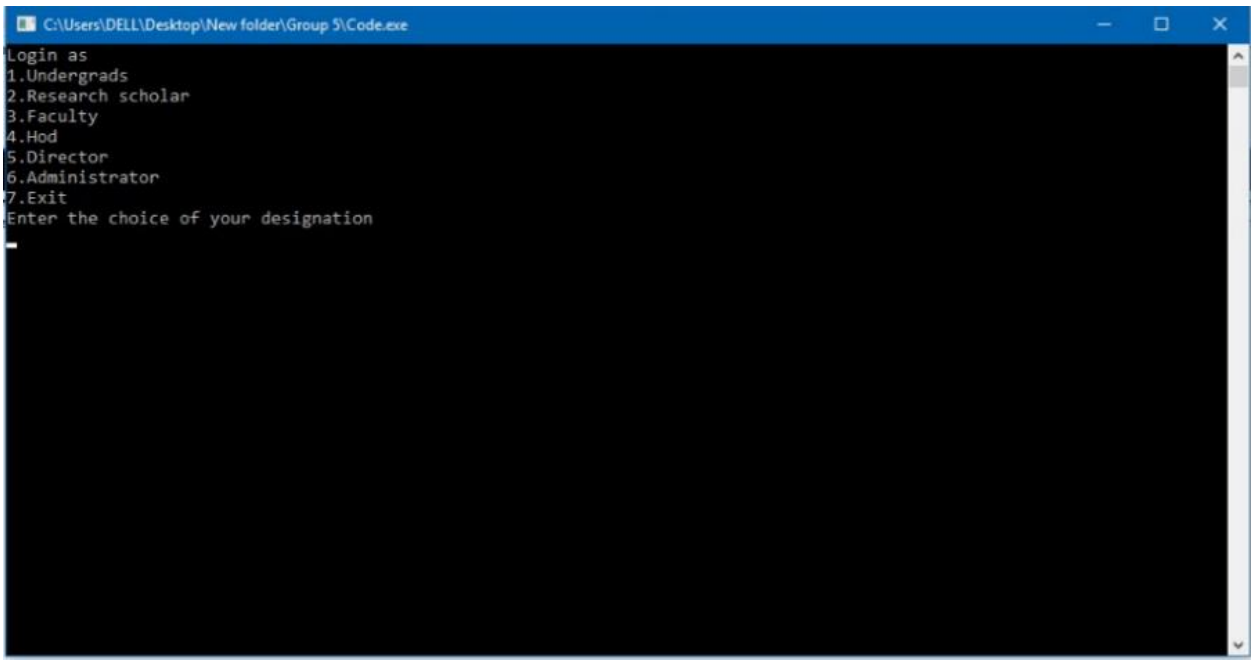

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Register as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Exit
Enter the choice of your designation
2
Enter name
Beta
Enter userid
2
enter family income
1800000
enter faculty id under which you are working
3
Concerned faculty not found..retry
Redirecting in
3

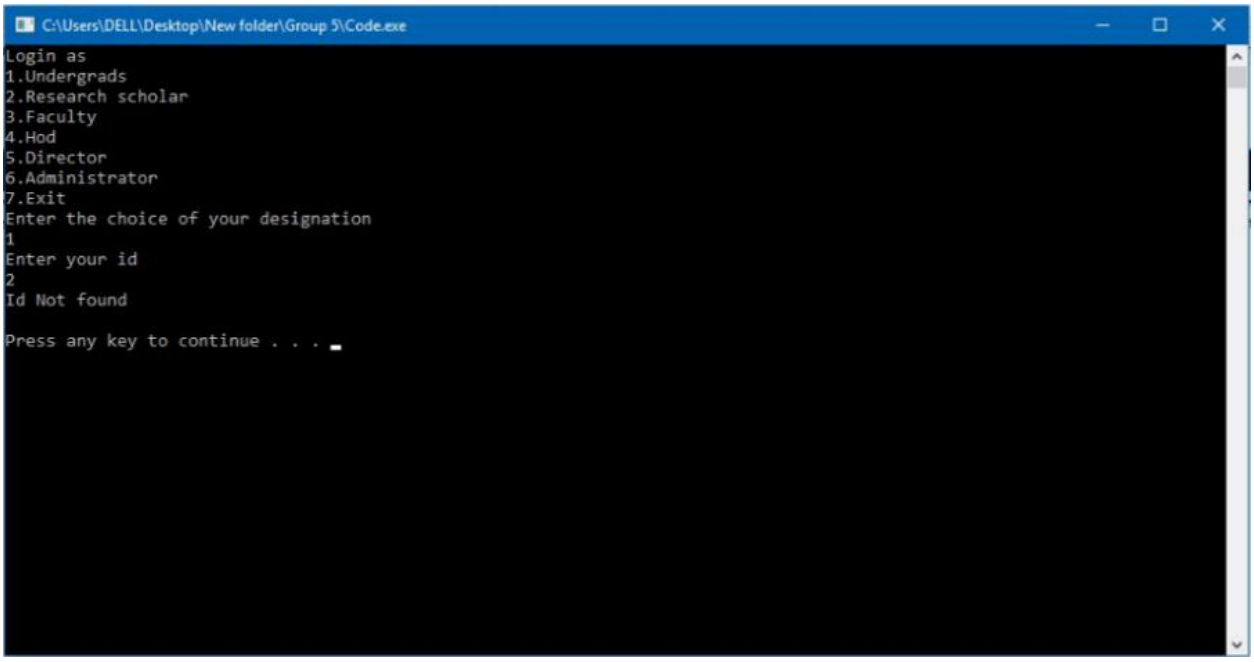

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Register as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Exit
Enter the choice of your designation
4
Enter name
Epsilon
Enter userid
5
Enter your department
CSE
Successfully registered!
Redirecting in
3

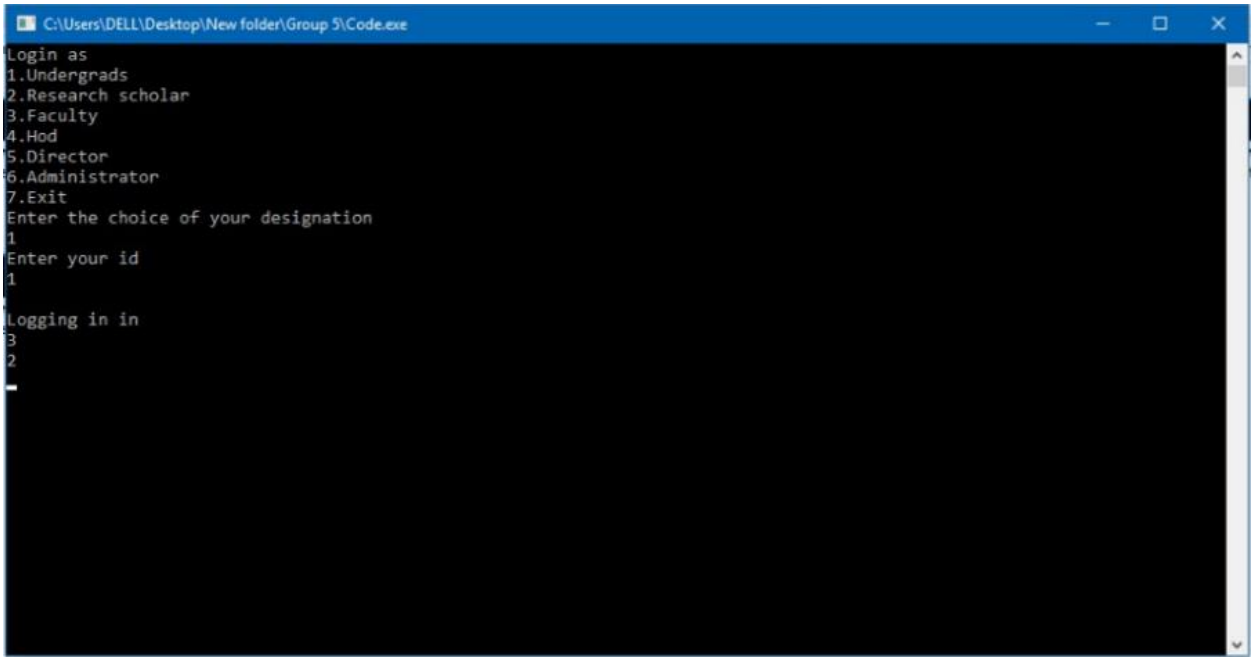

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Register as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Exit
Enter the choice of your designation
2
Enter name
Beta
Enter userid
2
enter family income
1800000
enter faculty id under which you are working
3
Successfully registered!
Redirecting in
3

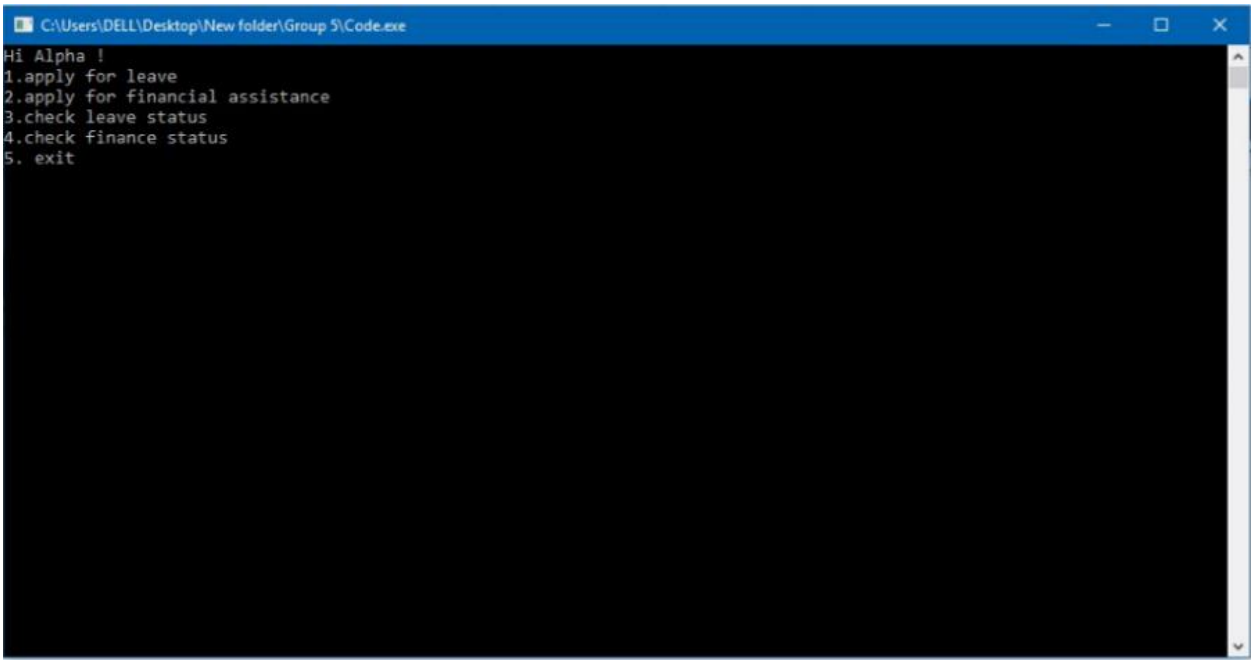

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Register as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Exit
Enter the choice of your designation
3
Enter name
Gamma
Enter userid
3
Enter your department
CSE
Successfully registered!
Redirecting in
3
2

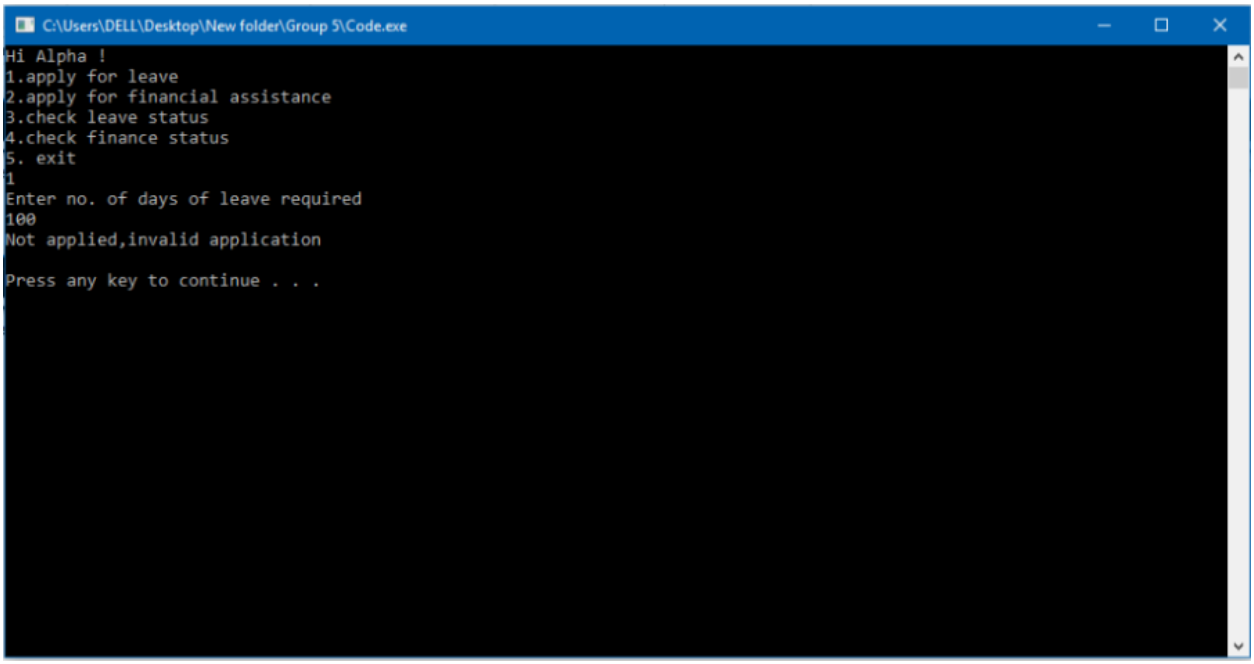

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Welcome back! What would you like to do?
1.Register
2.Login
3.exit
2
Directing for Login in
3
2

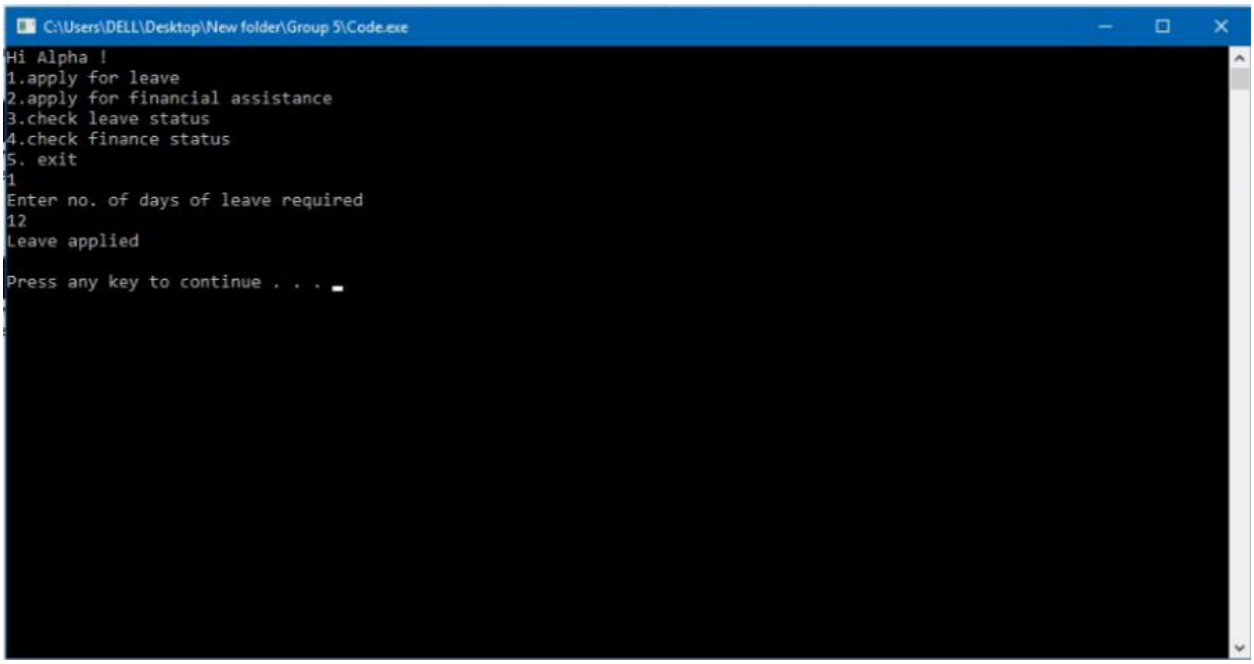

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation

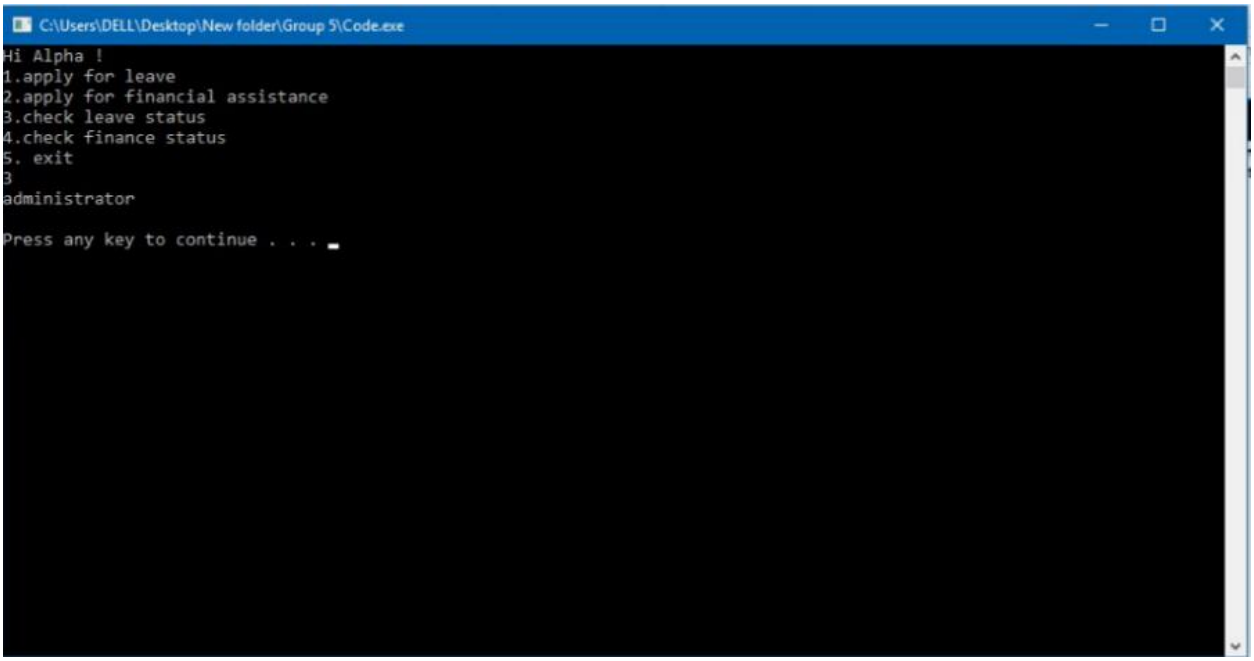

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
1
Enter your id
2
Id Not found
Press any key to continue . . .

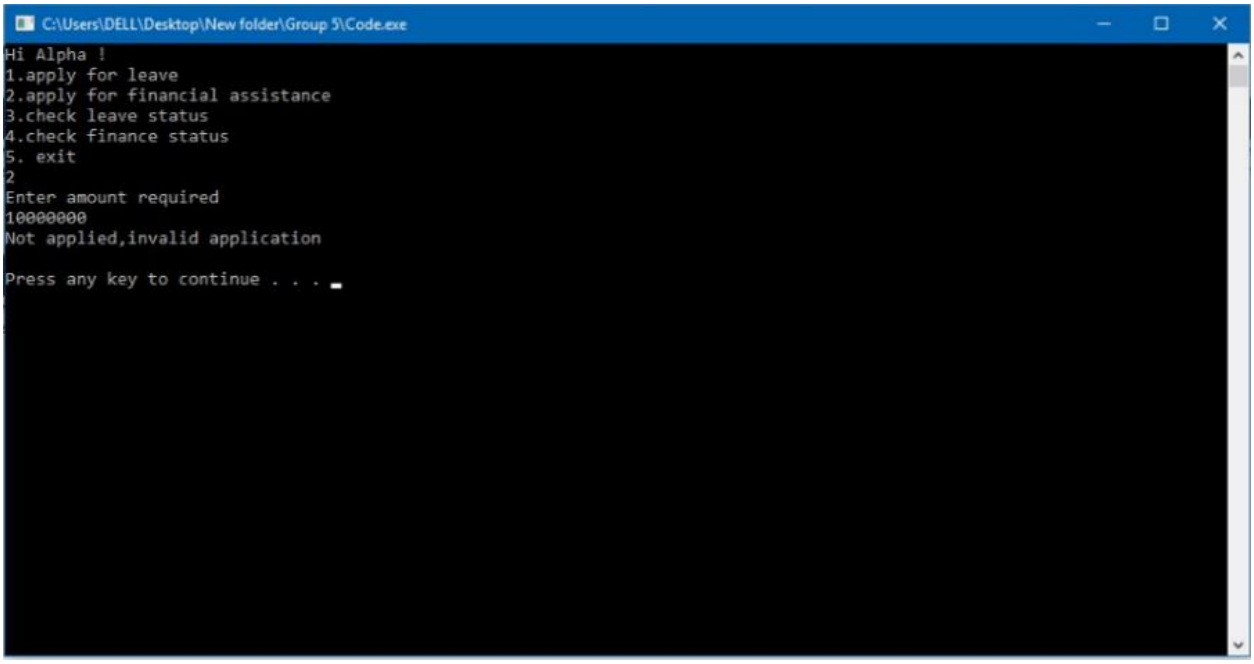

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
1
Enter your id
1
Logging in in
3
2

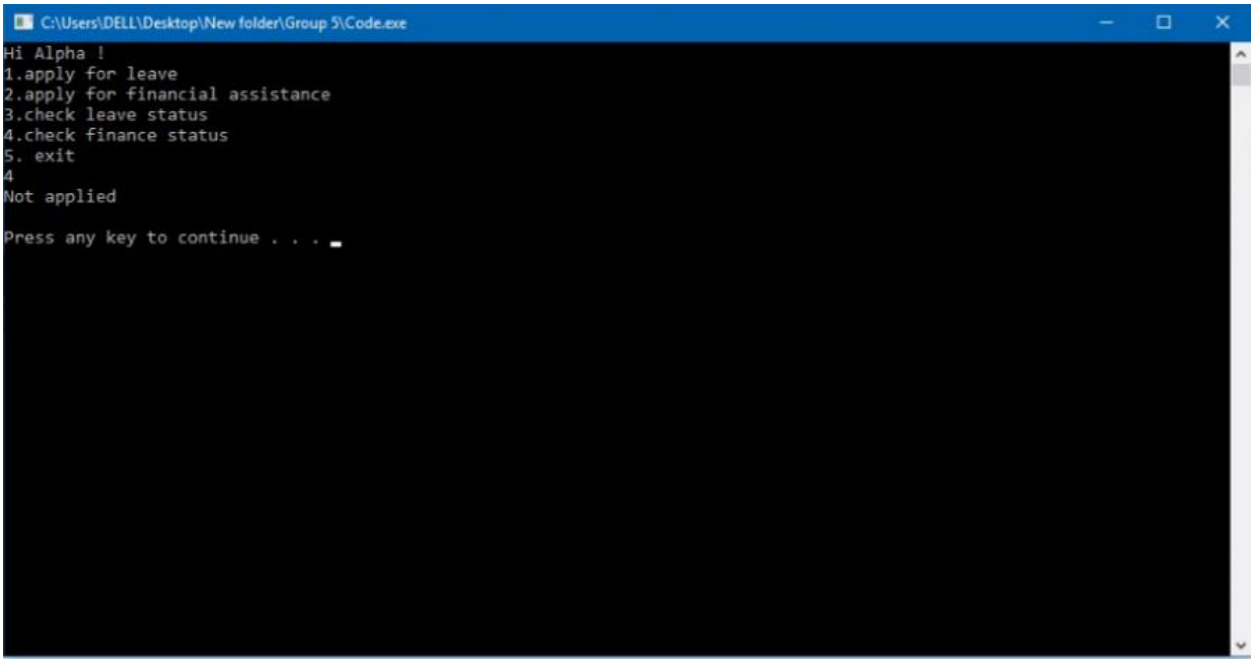

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit

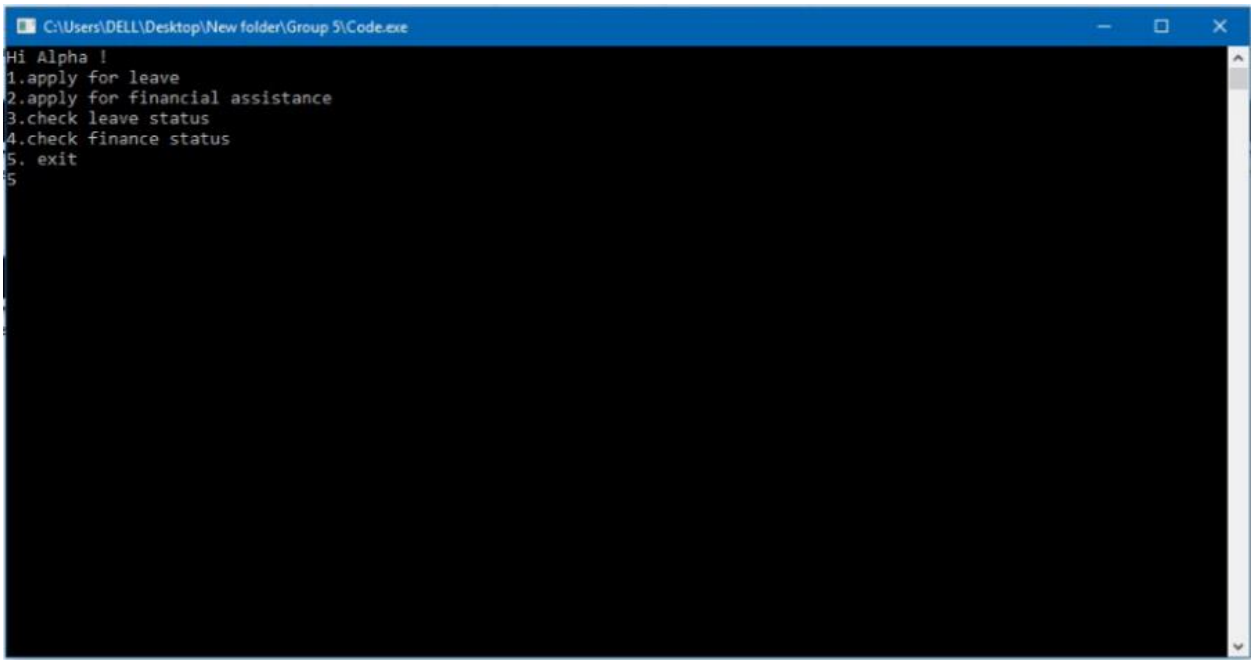

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
1
Enter no. of days of leave required
100
Not applied,invalid application
Press any key to continue . . .

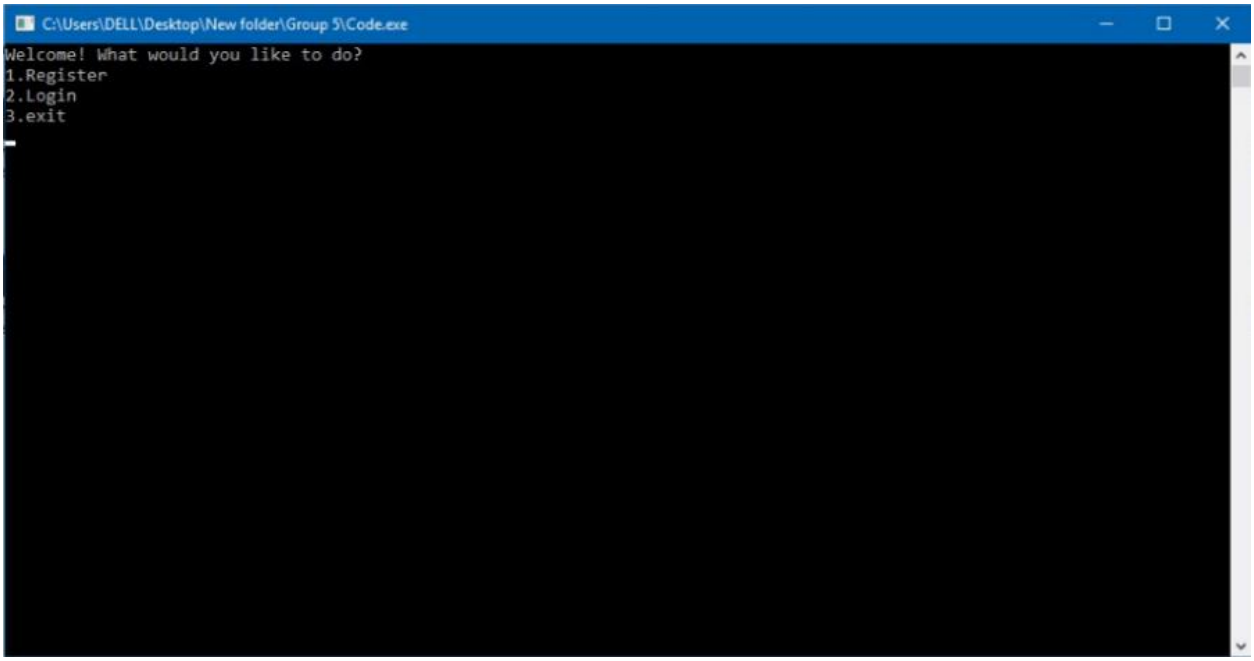

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
1
Enter no. of days of leave required
12
Leave applied
Press any key to continue . . .

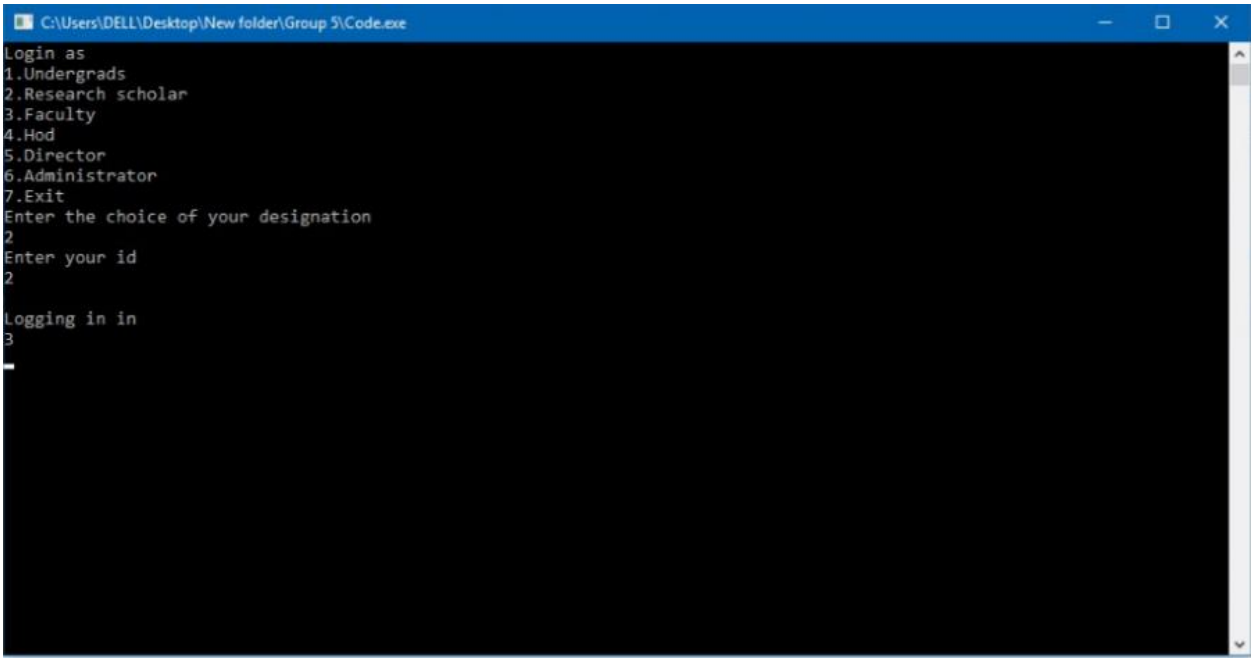

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
3
administrator
Press any key to continue . . .

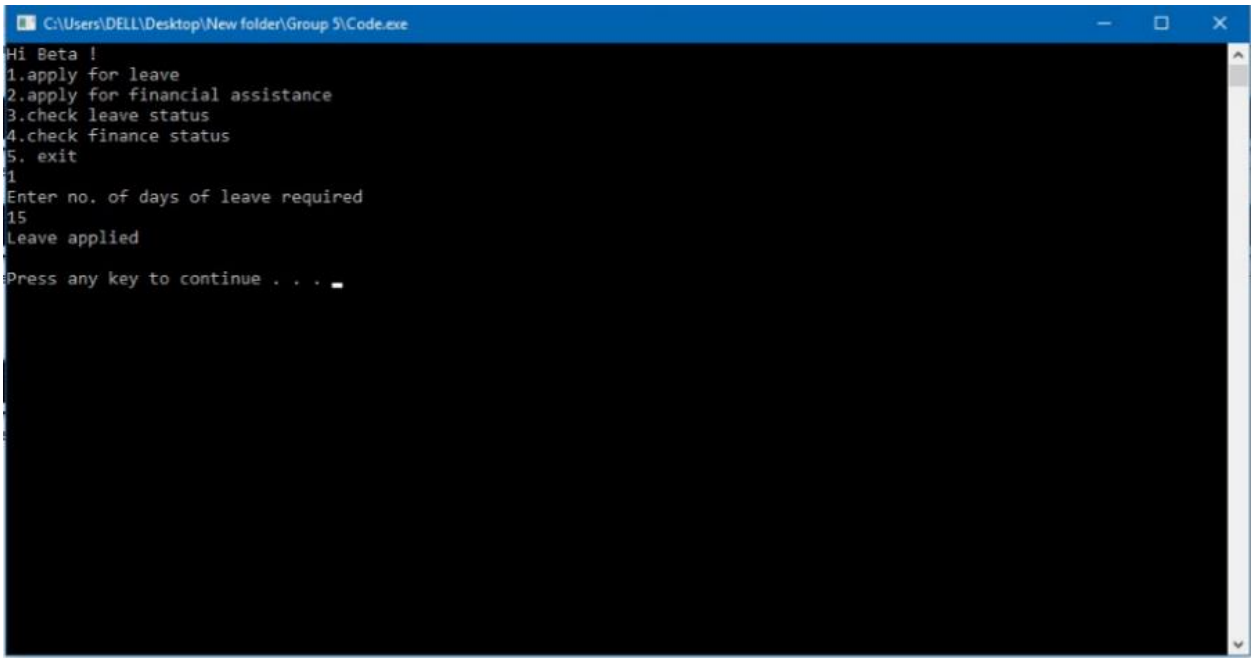

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
2
Enter amount required
10000000
Not applied,invalid application
Press any key to continue . . .

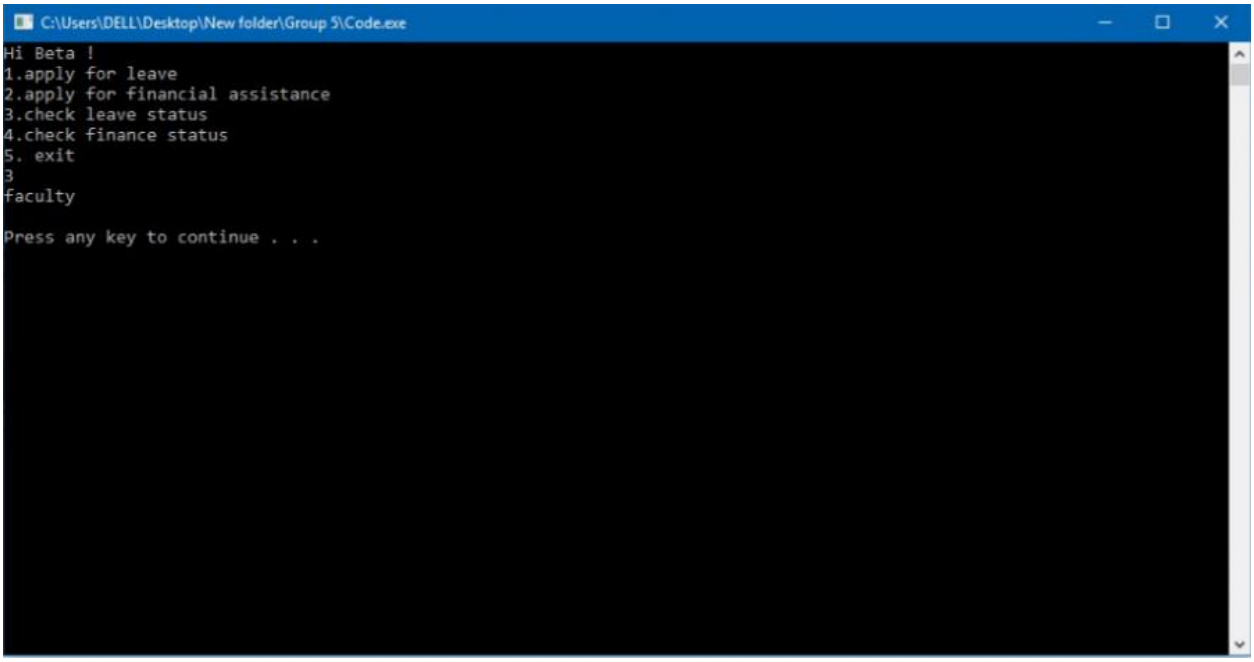

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
4
Not applied
Press any key to continue . . .

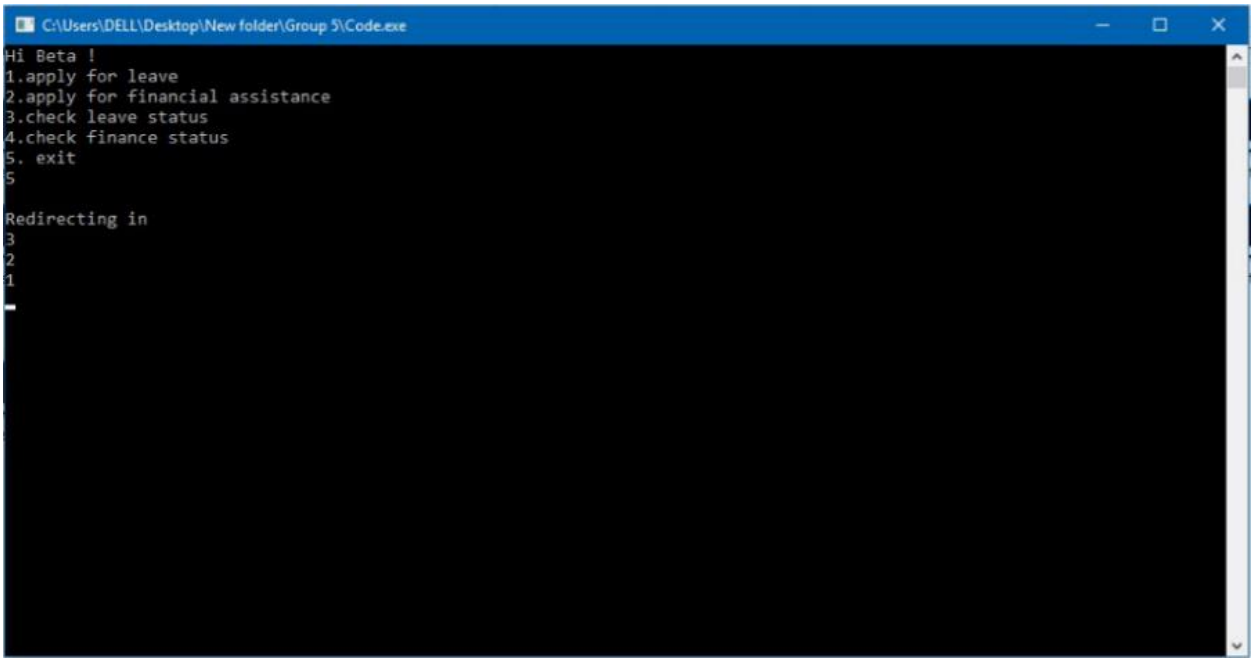

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
5

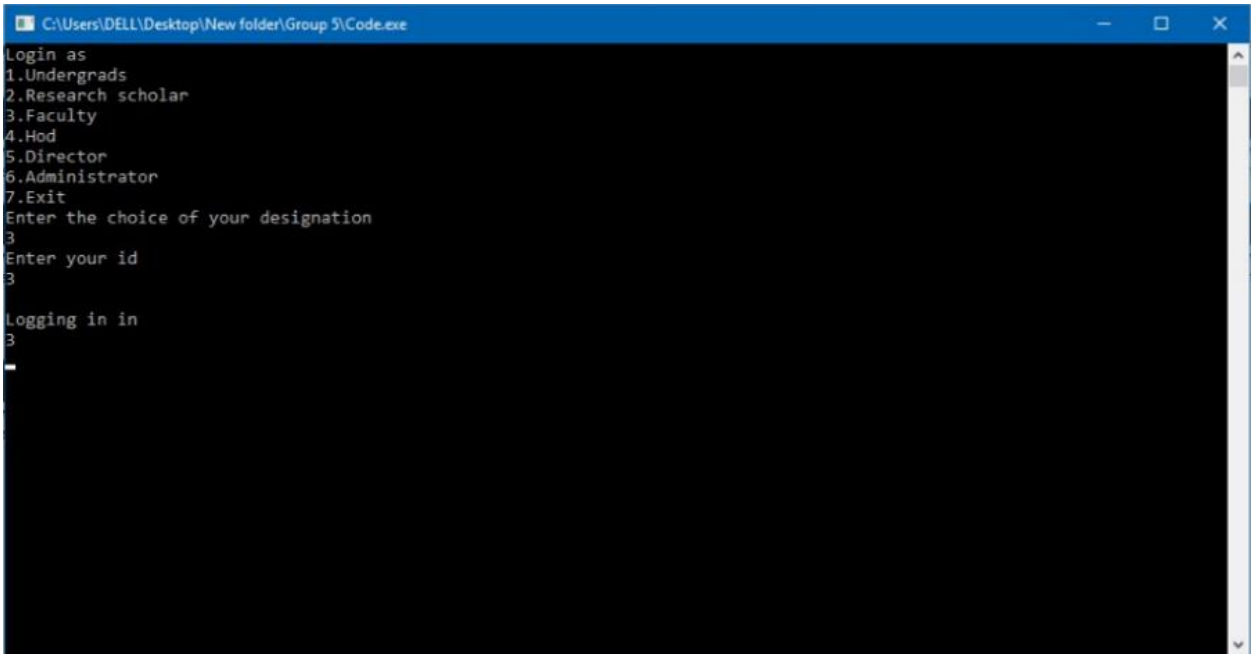

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Welcome! What would you like to do?
1.Register
2.Login
3.exit

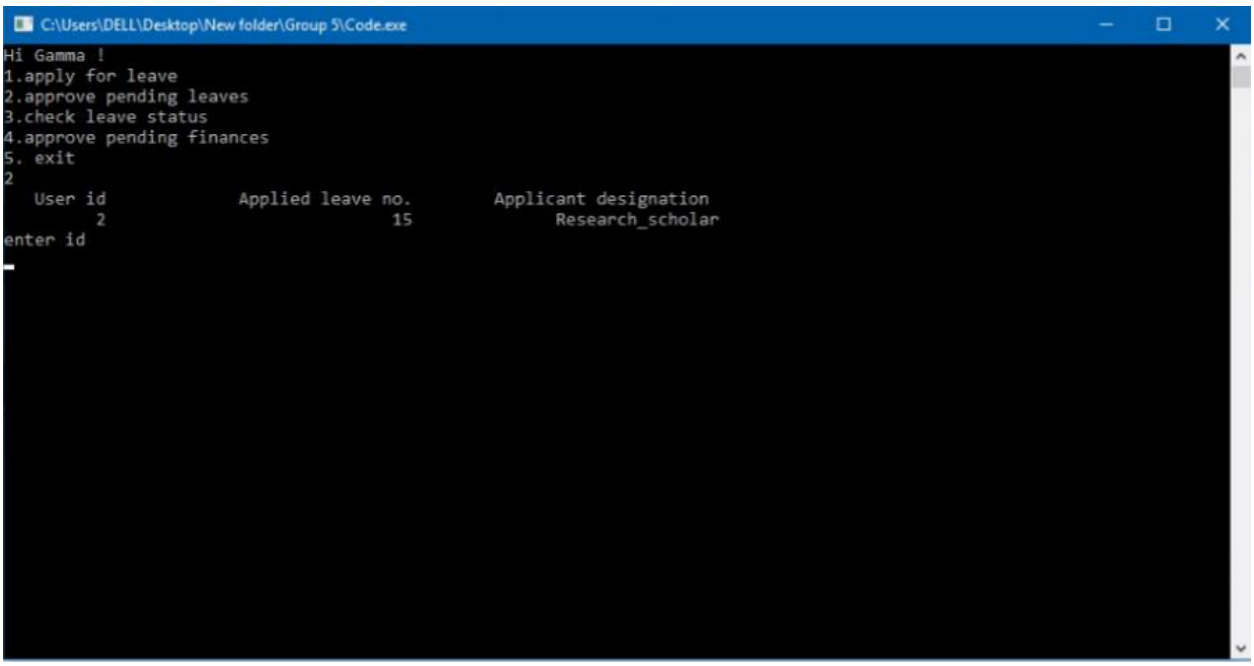

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
2
Enter your id
2
Logging in in
3

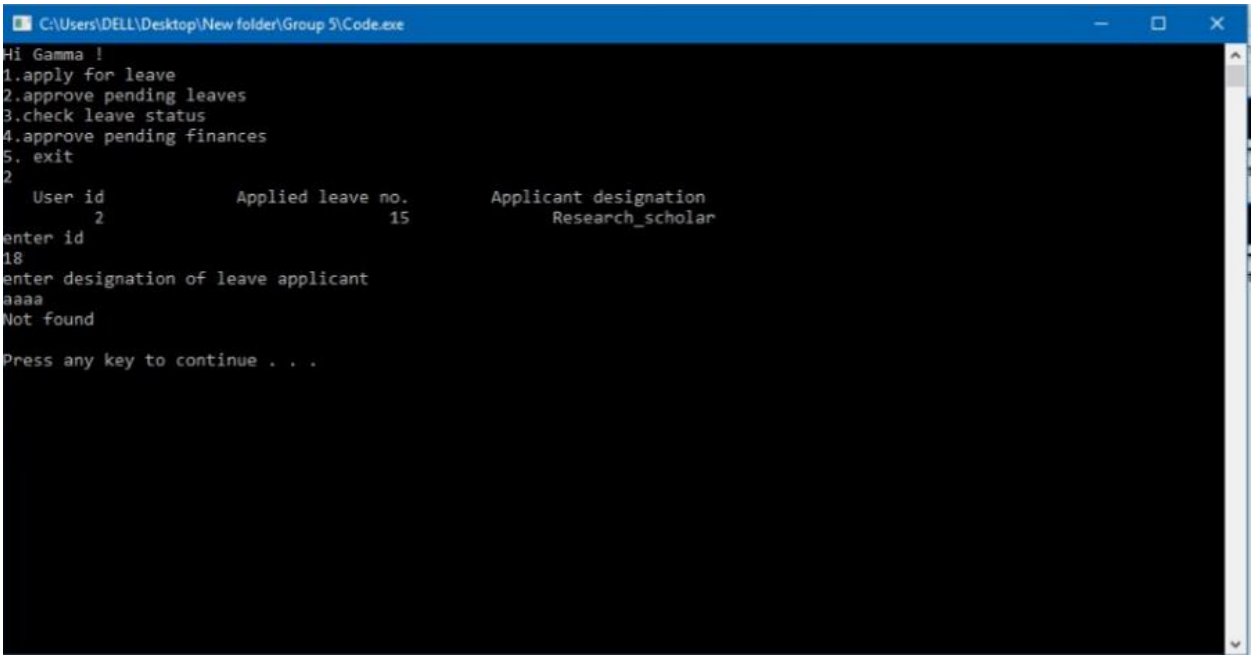

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Beta !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
1
Enter no. of days of leave required
15
Leave applied
Press any key to continue . . .

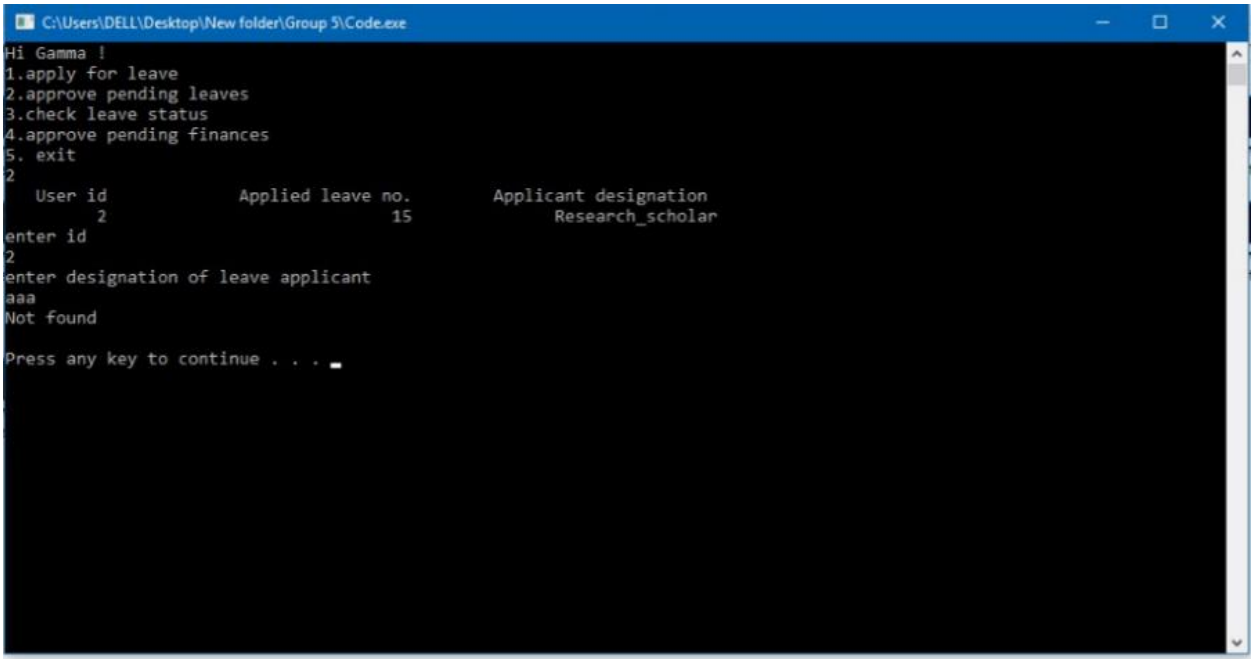

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Beta !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
3
faculty
Press any key to continue . . .

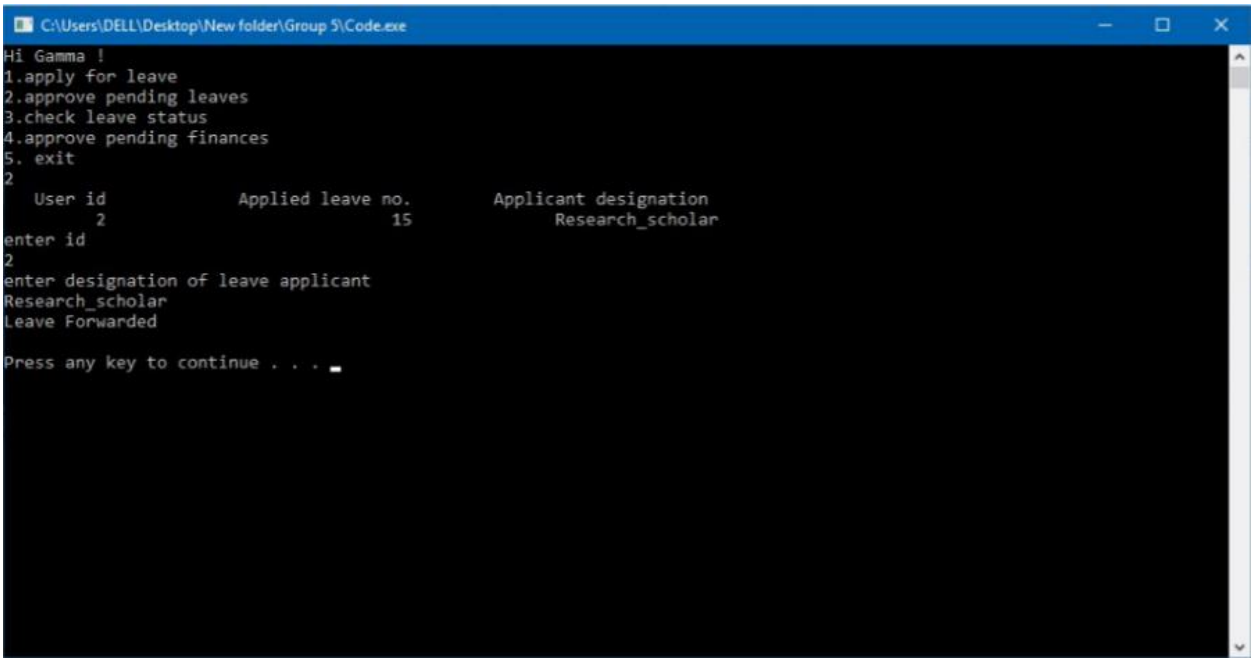

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Beta !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
5
Redirecting in
3
2
1

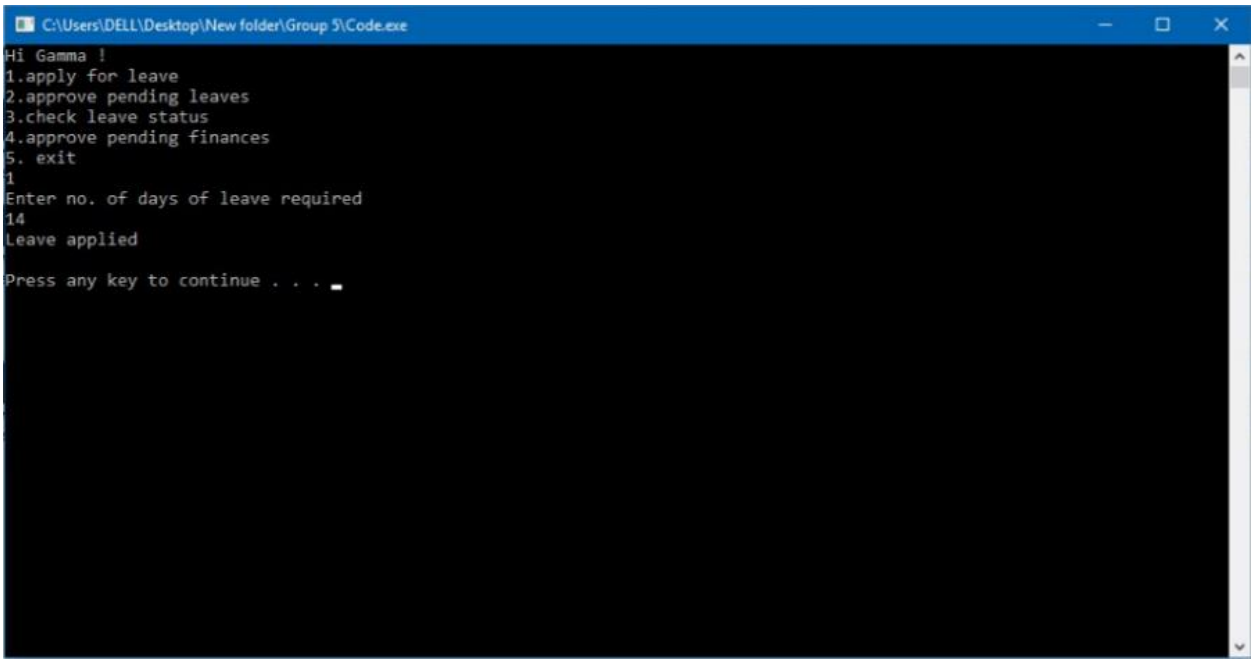

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
3
Enter your id
3
Logging in in
3

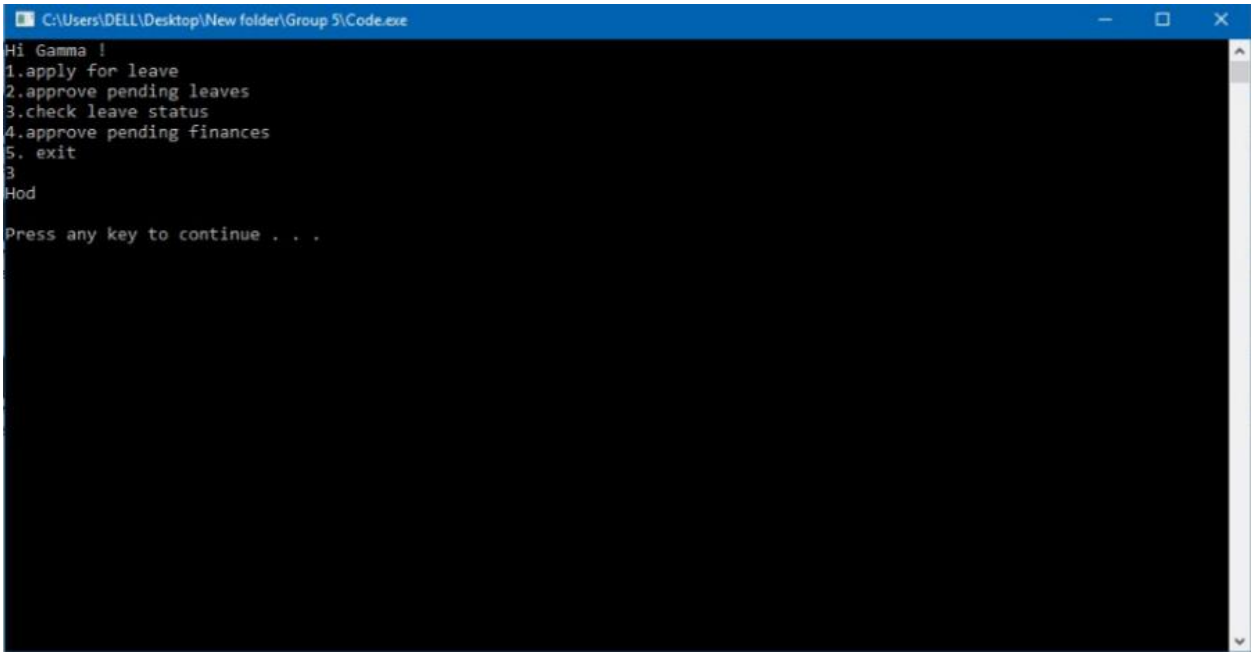

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
2
User id Applied leave no. Applicant designation
2 15 Research_scholar
enter id

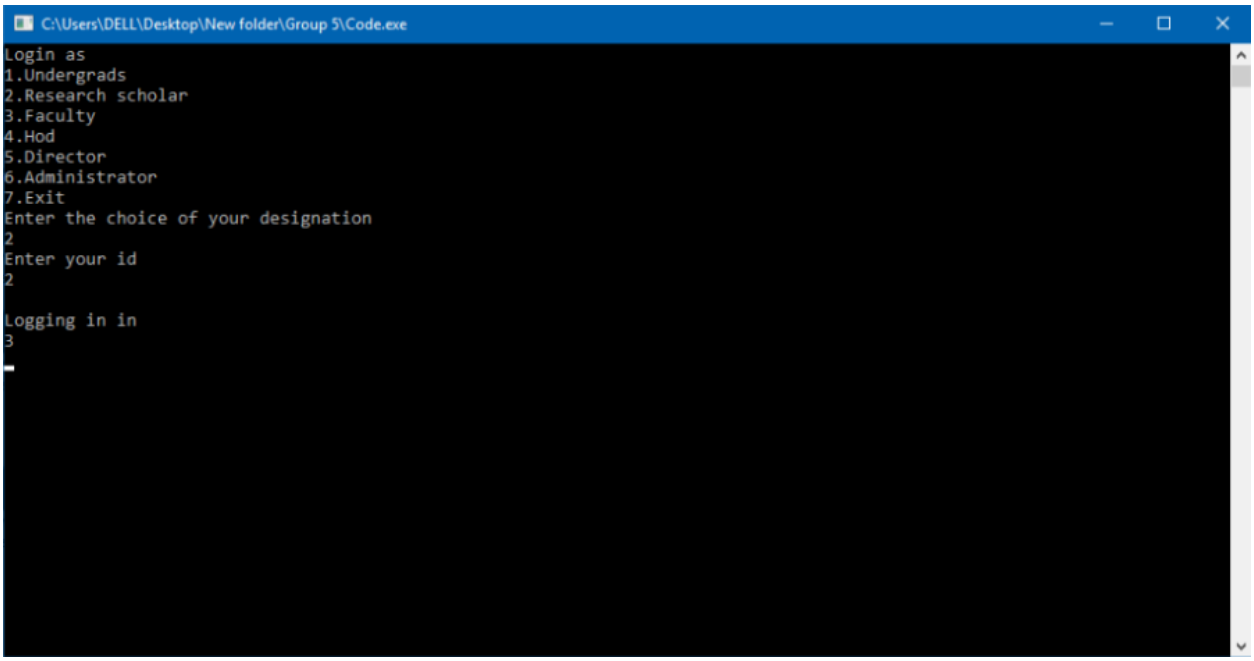

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
2
User id Applied leave no. Applicant designation
2 15 Research_scholar
enter id
18
enter designation of leave applicant
aaaa
Not found
Press any key to continue . . .

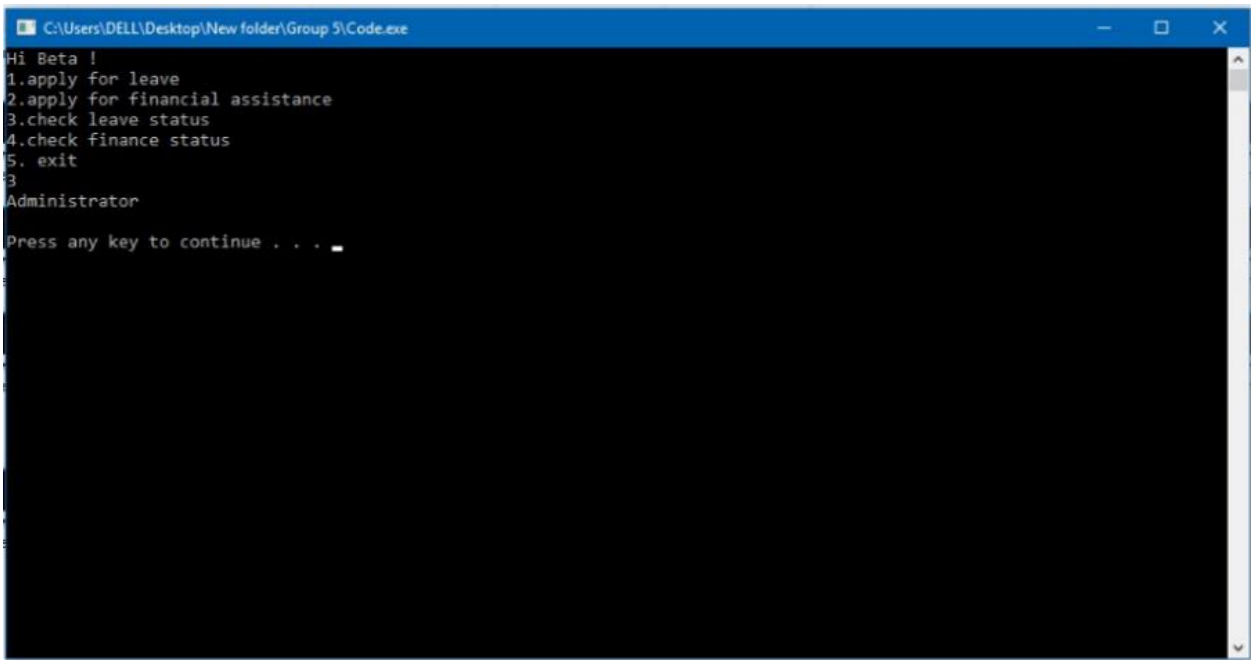

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
2
User id Applied leave no. Applicant designation
2 15 Research_scholar
enter id
2
enter designation of leave applicant
aaa
Not found
Press any key to continue . . .

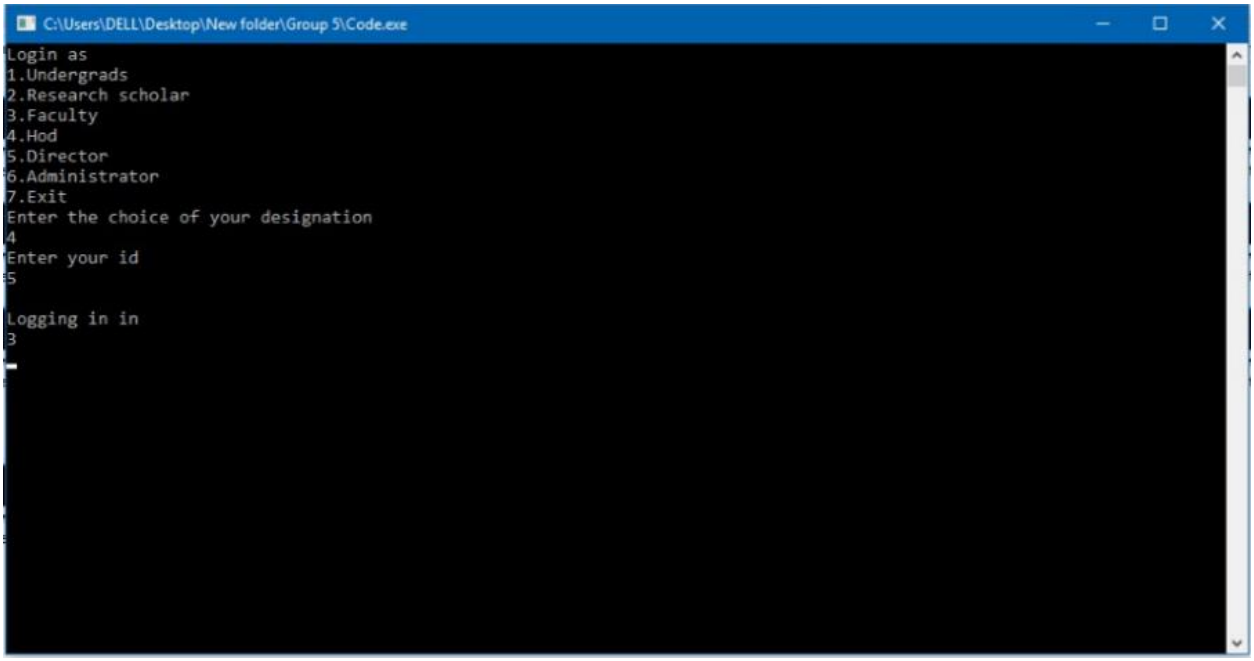

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
2
User id Applied leave no. Applicant designation
2 15 Research_scholar
enter id
2
enter designation of leave applicant
Research_scholar
Leave Forwarded
Press any key to continue . . .

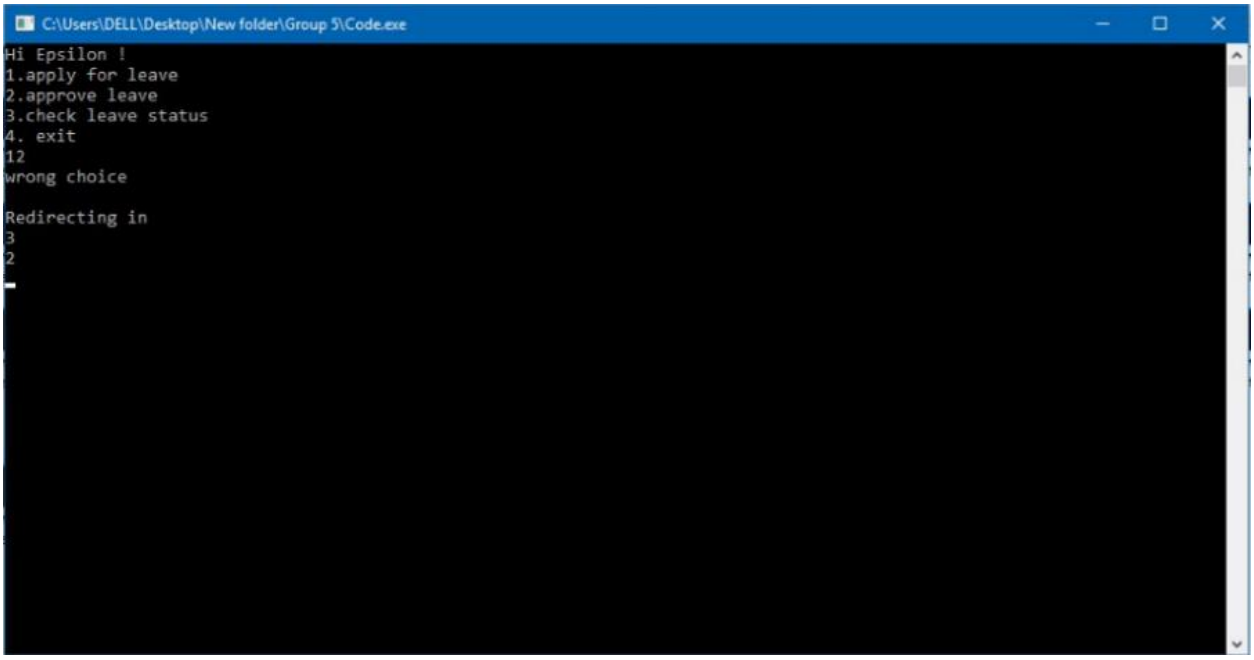

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
1
Enter no. of days of leave required
14
Leave applied
Press any key to continue . . .

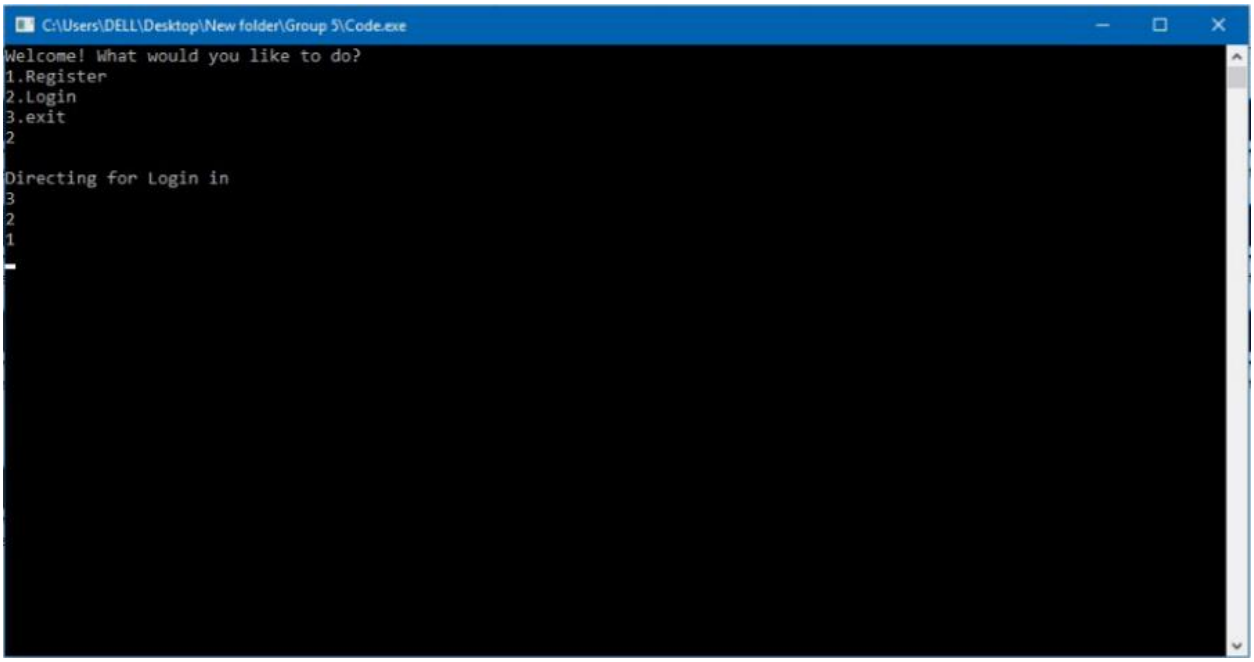

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
3
Hod
Press any key to continue . . .

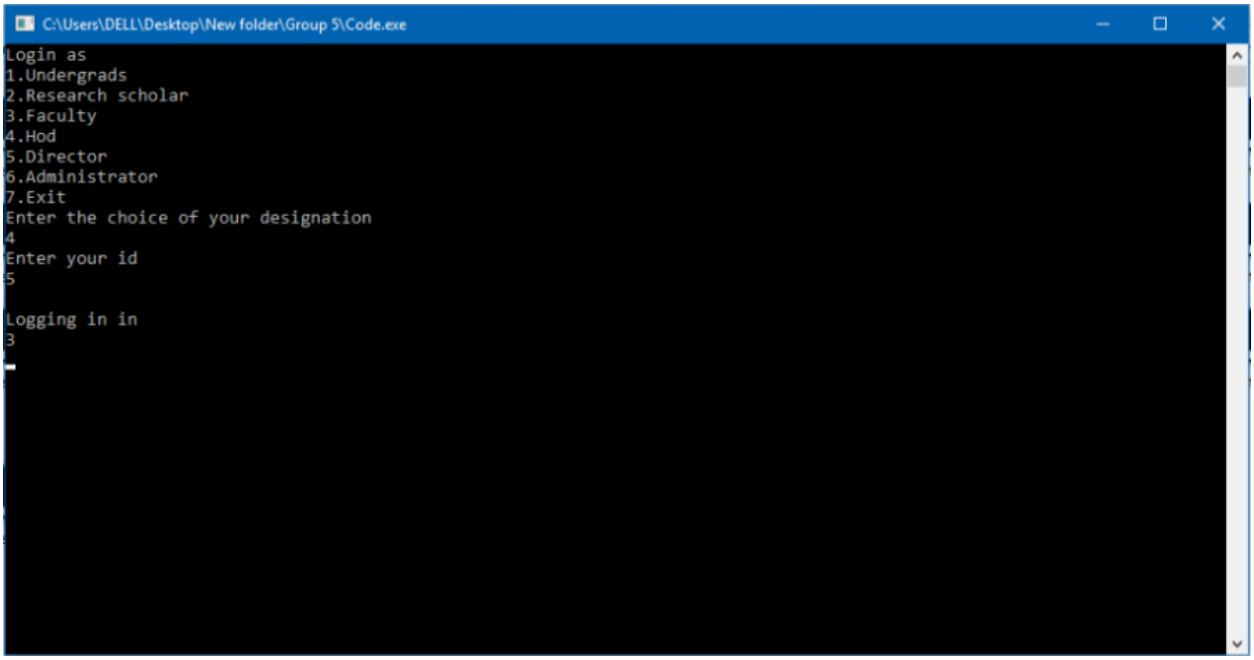

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
2
Enter your id
2
Logging in in
3

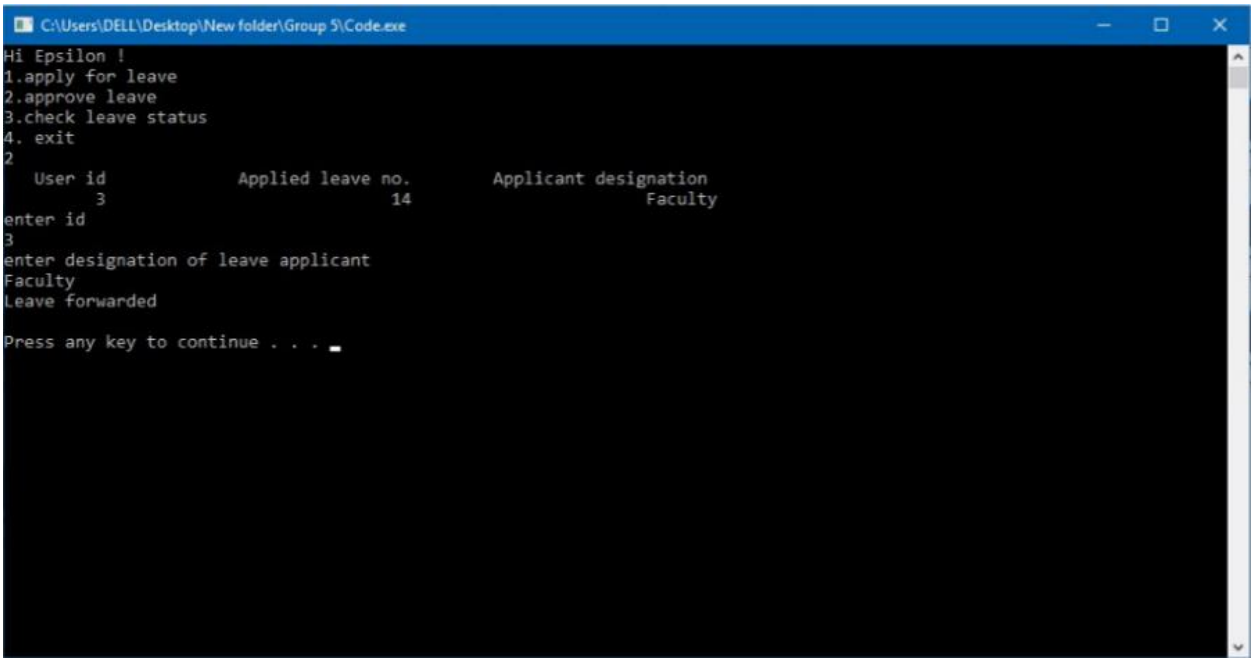

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Beta !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
3
Administrator
Press any key to continue . . .

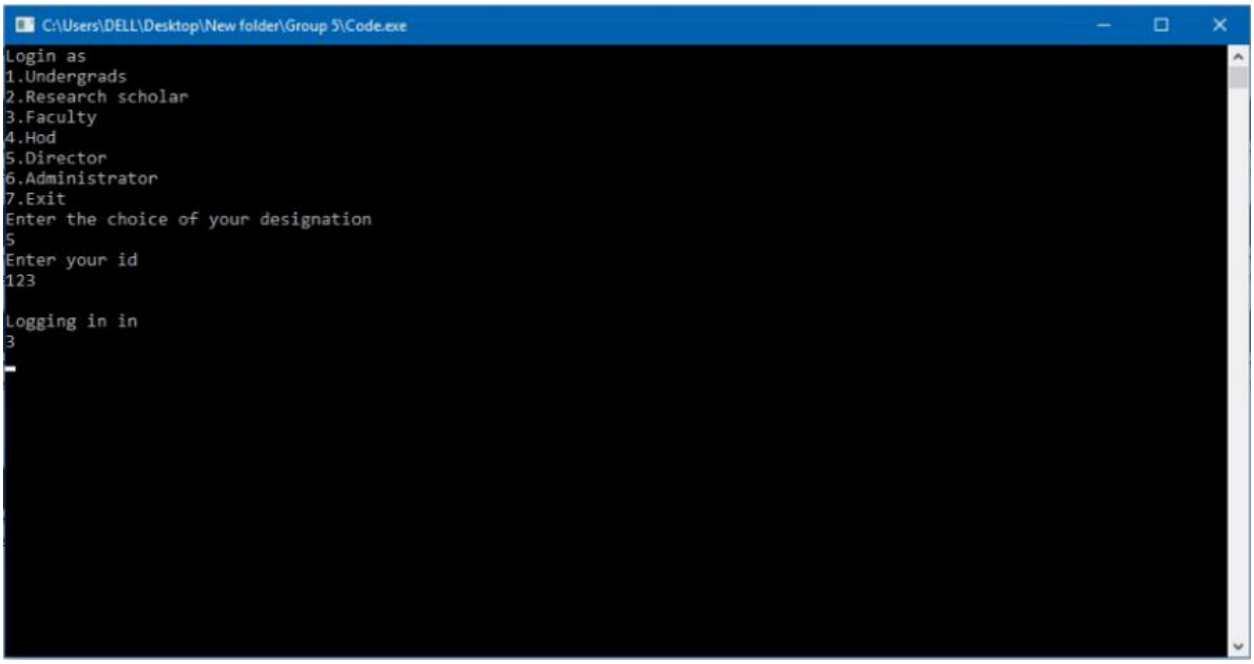

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
4
Enter your id
5
Logging in in
3

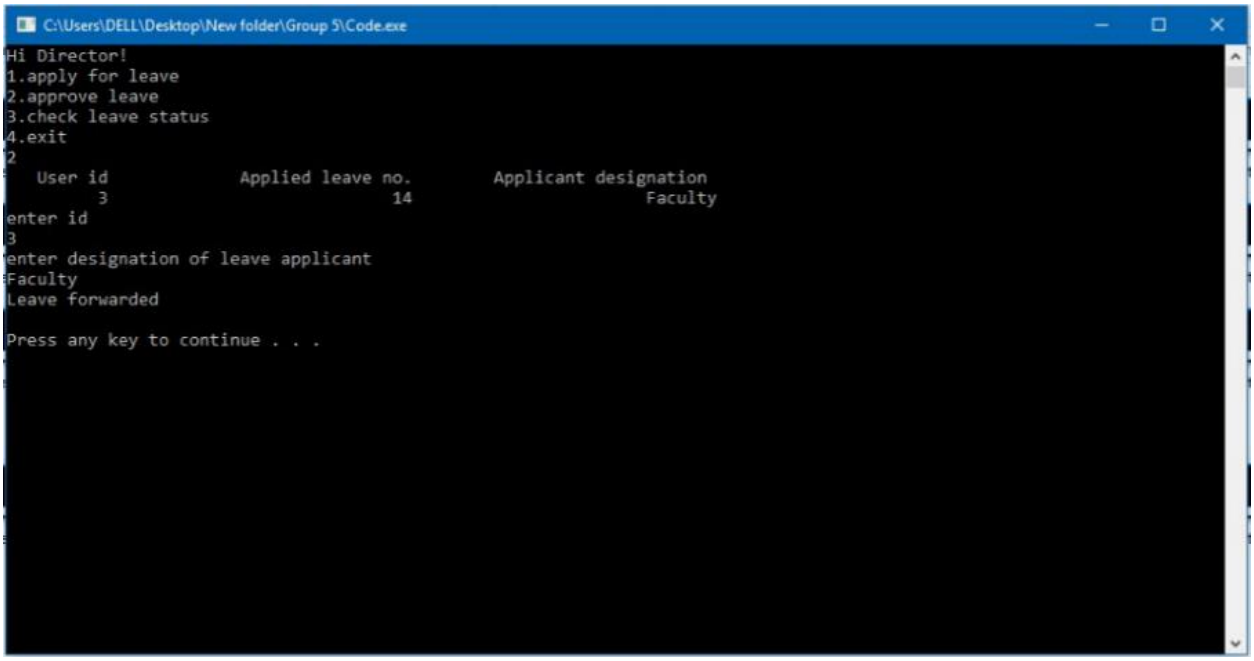

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Epsilon !
1.apply for leave
2.approve leave
3.check leave status
4. exit
12
wrong choice
Redirecting in
3
2

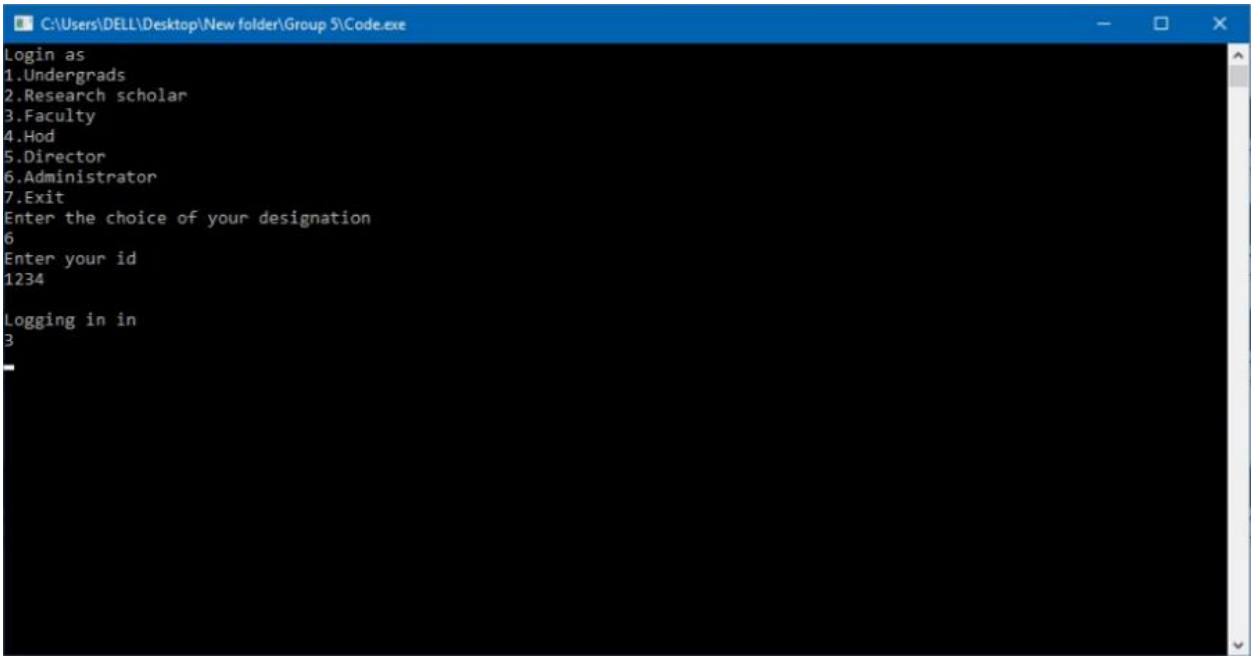

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Welcome! What would you like to do?
1.Register
2.Login
3.exit
2
Directing for Login in
3
2
1

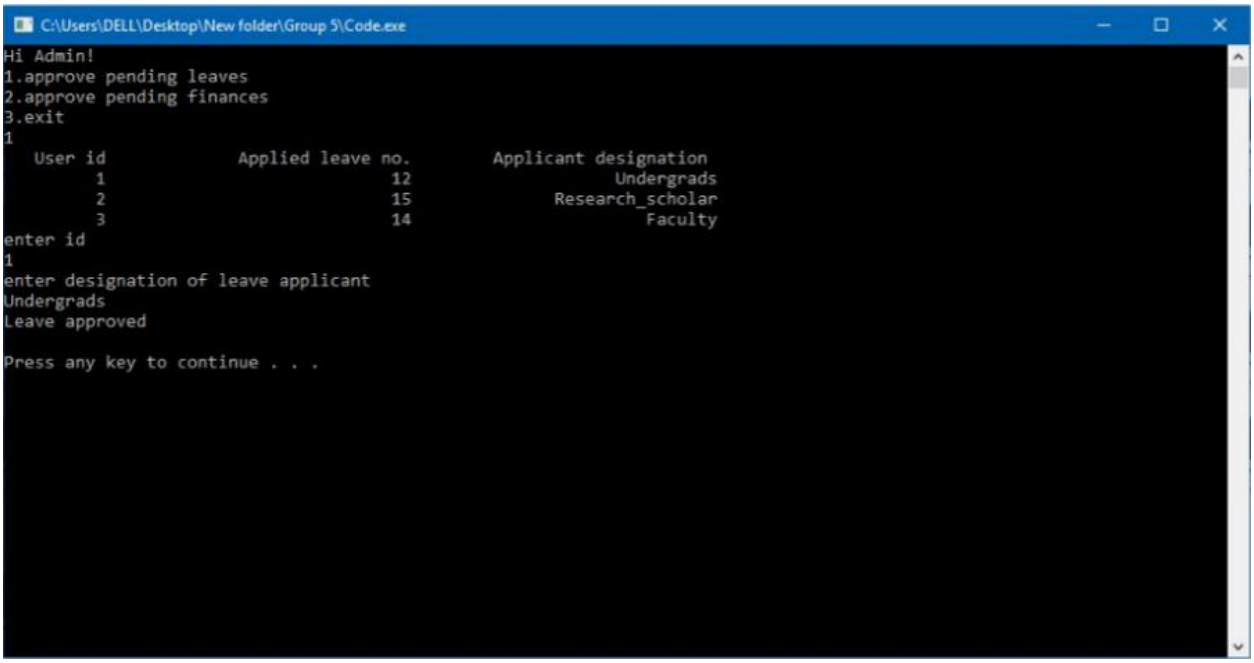

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
4
Enter your id
5
Logging in in
3

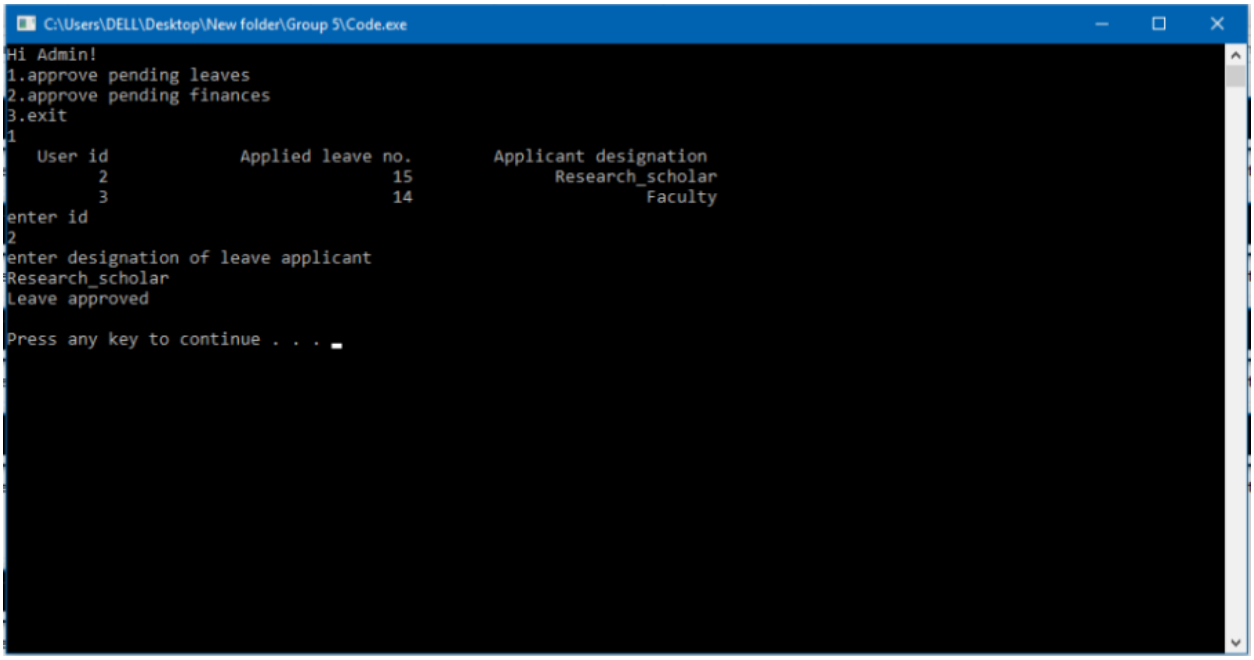

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Epsilon !
1.apply for leave
2.approve leave
3.check leave status
4. exit
2
User id Applied leave no. Applicant designation
3 14 Faculty
enter id
3
enter designation of leave applicant
Faculty
Leave forwarded
Press any key to continue . . .

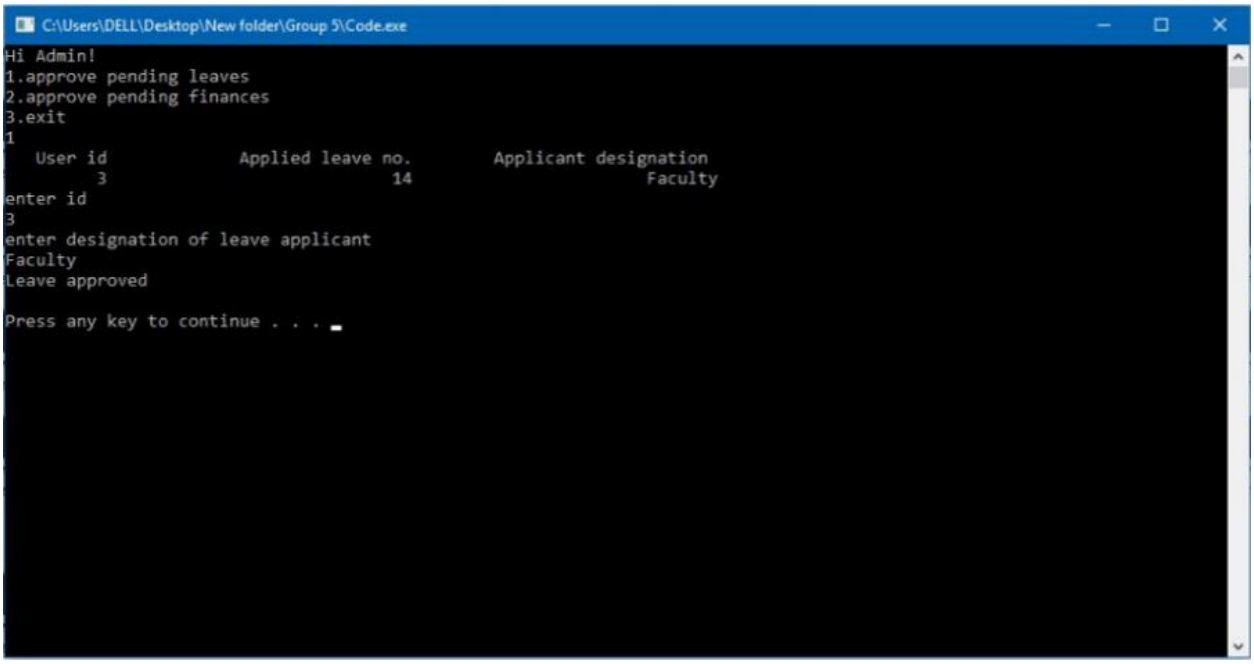

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
5
Enter your id
123
Logging in in
3

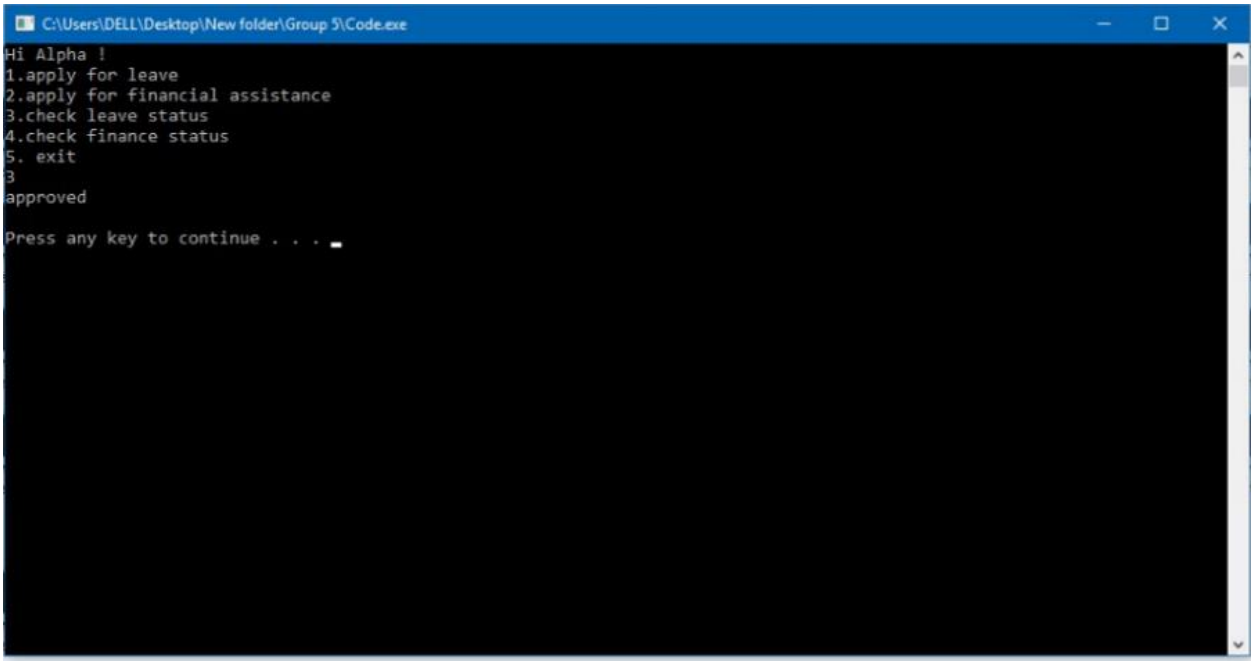

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Director!
1.apply for leave
2.approve leave
3.check leave status
4.exit
2
User id Applied leave no. Applicant designation
3 14 Faculty
enter id
3
enter designation of leave applicant
Faculty
Leave forwarded
Press any key to continue . . .

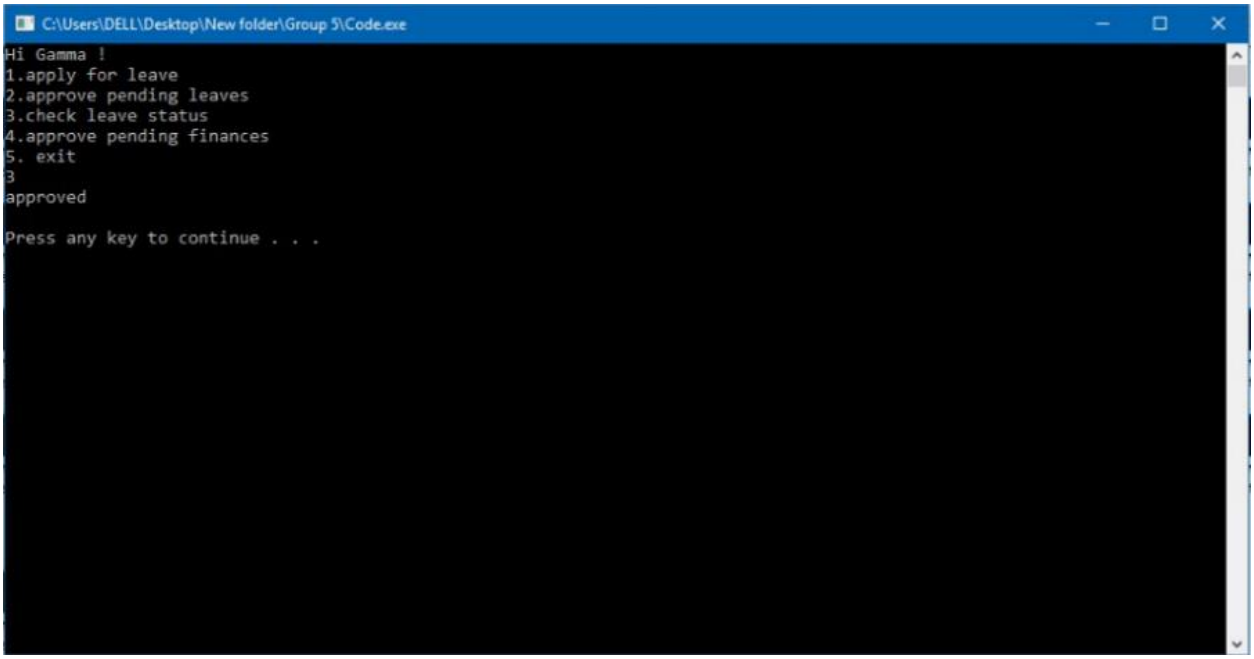

C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Login as
1.Undergrads
2.Research scholar
3.Faculty
4.Hod
5.Director
6.Administrator
7.Exit
Enter the choice of your designation
6
Enter your id
1234
Logging in in
3


C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Admin!
1.approve pending leaves
2.approve pending finances
3.exit
1
User id Applied leave no. Applicant designation
1 12 Undergrads
2 15 Research_scholar
3 14 Faculty
enter id
1
enter designation of leave applicant
Undergrads
Leave approved
Press any key to continue . . .


C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Admin!
1.approve pending leaves
2.approve pending finances
3.exit
1
User id Applied leave no. Applicant designation
2 15 Research_scholar
3 14 Faculty
enter id
2
enter designation of leave applicant
Research_scholar
Leave approved
Press any key to continue . . .


C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Admin!
1.approve pending leaves
2.approve pending finances
3.exit
1
User id Applied leave no. Applicant designation
3 14 Faculty
enter id
3
enter designation of leave applicant
Faculty
Leave approved
Press any key to continue . . .


C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Alpha !
1.apply for leave
2.apply for financial assistance
3.check leave status
4.check finance status
5. exit
3
approved
Press any key to continue . . .


C:\Users\DELL\Desktop\New folder\Group 5\Code.exe
Hi Gamma !
1.apply for leave
2.approve pending leaves
3.check leave status
4.approve pending finances
5. exit
3
approved
Press any key to continue . . .